

Apress®

Pro Linux
System Administration

Linux 系统管理大全

James Turnbull
[美] Peter Lieverdink 著
Dennis Matotek

张铮 张勇 陈勇涛 刘春华 耿强 译

 人民邮电出版社
POSTS & TELECOM PRESS

Linux系统管理大全

亲爱的读者：

我们编写本书的目的是帮助中小型企业破除商业软件的束缚，并向他们说明实施备选的自由软件方案是多么容易。本书阐述了如何使用Linux和开源软件帮助企业更好地控制他们的技术方向并降低成本。

我们向您介绍如何实施与管理Linux服务器、Linux服务和应用程序，并演示了管理单位的IT服务是多么容易。通过本书，读者可以获知如何安装与管理重要的商业工具，如自己的E-mail服务器与Web服务器；如何实施如文档管理、文件服务与打印这样的服务；如何使用包括E-mail与日程表在内的完整协作程序组。读者还将学习到如网络连接、事件记录、备份和配置管理这样的配套服务，所有这些都有助于管理您的平台。

我们采用积木式的方法逐步向您讲述如何创建Linux基础构架，以及如何把业务转移到自由软件与开源软件上来。本书从安装第一台Linux服务器开始，接着介绍Linux基础知识，包括安装和配置第一批Linux应用程序，再到更高级的概念，如大型服务器管理和虚拟化技术。本书结束时，您将有望成为Linux专家，并具备管理自己的Linux服务器所需要的一切技巧和知识。

James Turnbull、Peter Lieverdink与Dennis Matotek

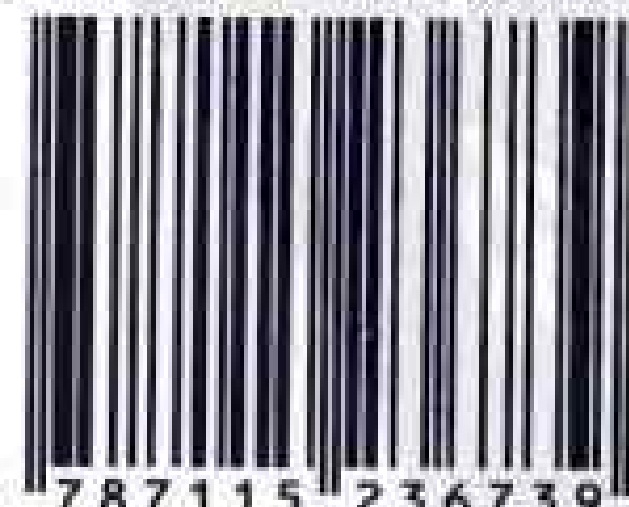


封面设计 胡萍丽

分类建议：计算机 / 操作系统 / Linux

人民邮电出版社网址：www.ptpress.com.cn

ISBN 978-7-115-23673-9



9 787115 236739 >

ISBN 978-7-115-23673-9

定价：108.00 元

Linux 系统管理大全

James Turnbull
[美] Peter Lieverdink 著
Dennis Matotek

张铮 张勇 陈勇涛 刘春华 耿强 译

图书在版编目 (C I P) 数据

Linux系统管理大全 / (美) 特恩布尔
(Turnbull, J.), (美) 利沃德林克 (Lieverdink, P.),
(美) 马托泰克 (Matotek, D.) 著 ; 张铮等译. — 北京
: 人民邮电出版社, 2010.12
ISBN 978-7-115-23673-9

I. ①L… II. ①特… ②利… ③马… ④张… III. ①
Linux操作系统 IV. ①TP316.89

中国版本图书馆CIP数据核字(2010)第167924号

版 权 声 明

Original English language edition, entitled Pro Linux System Administration by James Turnbull, Peter Lieverdink, Dennis Matotek, published by Apress 2855 Telegraph Avenue, #600, Berkeley, CA 94705 USA.
Copyright © 2009 by Apress L.P. Simplified Chinese-language edition copyright © 2010 by POSTS & TELECOMMUNICATIONS PRESS.
All rights reserved.
本书中文简体字版由 Apress L.P.授权人民邮电出版社独家出版。未经出版者书面许可,不得以任何方式复制或抄袭本书内容。
版权所有,侵权必究。

Linux 系统管理大全

- ◆ 著 [美] James Turnbull Peter Lieverdink Dennis Matotek
译 张 铮 张 勇 陈勇涛 刘春华 耿 强
责任编辑 傅道坤
- ◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子函件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
北京隆昌伟业印刷有限公司印刷
- ◆ 开本: 787×1092 1/16
印张: 52.5
字数: 1256 千字 2010 年 12 月第 1 版
印数: 1-3 000 册 2010 年 12 月北京第 1 次印刷
著作权合同登记号 图字: 01-2009-2888 号
ISBN 978-7-115-23673-9

定价: 108.00 元

读者服务热线: (010)67132705 印装质量热线: (010)67129223
反盗版热线: (010)67171154

内 容 提 要

本书介绍了 Linux 操作系统的安装、服务配置和系统管理的相关知识和应用技巧。全书共分为两个部分：第一部分以具有广泛代表性的 Red Hat Enterprise Linux 和 Ubuntu 为蓝本，介绍了如何安装、配置和管理 Linux 系统等基础的 Linux 知识；第二部分通过实际安装和配置诸如邮件服务、网站服务、文件和打印机共享服务、目录服务等一些企业运营所需的服务，展示了如何将学到的各种 Linux 技术应用于实际，同时本部分还介绍了系统安装和服务管理的自动化技术。

本书是一本不可多得的 Linux 应用参考指南，本书的三位作者都是 Linux 和开源领域的资深专家，他们结合自己的实际经验深入浅出地讲解，读者无需任何 Linux 基础就可以轻松掌握 Linux 服务器的安装和配置，并了解在服务器上运行的应用程序的大量相关知识。无论您是普通的开源爱好者，还是专业的 Linux 系统管理员，都可以从本书中获益。

献给孜孜不倦使一切都变得有价值的 Ruth；同时献给一直支持我的家人。

——James Turnbull

献给 Donna, Pixel 和 Mustafa。

——Peter Lieverdink

献给 Bianca 和我的孩子——Ziggy 和 Anika，以及我们的宠物。

——Dennis Matotek

关于作者

James Turnbull 领导澳大利亚国家银行 (National Australia Bank) 的计算机紧急反应小组 (Computer Emergency Response Team, CRET)。他同时是澳大利亚 Linux 联盟 (Linux Australia) 的一员, 包括 2008 年在执行委员会 (Executive Council) 和维多利亚 Linux 用户组 (Linux Users of Victoria) 委员会任职。

他是许多开源项目的贡献者, 并且经常针对写作、系统管理、开源技术等相关话题发表意见。他是以下 3 本书的作者。

《Pulling Strings with Puppet: Systems Administration Made Easy》, 该书探索基于 Ruby 的配置管理工具 Puppet。

《Hardening Linux》, 该书着眼于巩固 Linux 主机的安全, 内容包括底层操作系统、文件系统、防火墙、连接、日志、安全测试以及 E-mail、FTP 和 DNS 等大量普通应用程序的安全问题。

《Pro Nagios 2.0》, 该书介绍使用开源工具 Nagios 进行企业管理。

Peter Lieverdink 出生在荷兰的一个小乡镇。他有一双木屐。虽然他出生在荷兰, 但他几乎从不吃郁金香, 也没在风车房住过。

在他 22 岁生日那天, Peter 移居到澳大利亚, 并且很快就在一个办公室的隔断间工作。他现在有了自己的公司, Creative Contingencies Pty, Ltd。该公司依赖于开源软件, 为基础设施和建设以及日常的办公室工作服务。

Peter 专长于网络应用程序开发, 以及在桌面和服务器端使用 Linux 帮助其他企业实现开源解决方案。

Dennis Matotek 出生于澳大利亚维多利亚州的一个名叫 Mildura 的小镇。正如所有小镇一样, 在 Mildura, 物资的长期短缺驱使年轻一代到外面的世界去闯荡。随后, Dennis 移居到墨尔本。在那里, 亚拉河的波光倒映着整个城市的繁华。然而, 在爱尔兰, 经过两年为某个脾气暴躁、蓝面孔且身材矮小的苏格兰人卖力后, Dennis 最终被介绍从事系统管理工作。

在苏格兰, Dennis 工作于技术的边缘, 他有几台 486DX 的机器和一台 VAX 小型计算机。当回到墨尔本, 在机场 24 小时不眠地看管自己的行李后, Dennis 便很快得到一次面试机会——那个时候的工作机会就像天上的雪花一样多。

自从那个时候, Dennis 大部分时间都呆在墨尔本, 管理 IBM AS400 系列服务器 6 年, 并且主要在 Linux 系统方面工作了 9 年。Dennis 同时也著书, 而且执导和出演一些短片。他有一个可爱的妻子和一个名叫 Zigfryd 的儿子, 以及一个他在工作时每时每刻都在想念的刚出生的名叫 Anika 的女儿。

其实他从来没有遇到过那个苏格兰人。

关于技术评审员

Jaime Sicam 偶尔以 IT 领域的讲师和顾问的身份进行工作。在他全职工作之外的业余时间，他沉迷于安全防御者技术联盟（Defender Technologies Group）工程组中系统管理员的工作中。

Jaime 专长于 Bayanihan Linux，他以身为高级科学技术协会（Advanced Science and Technology Institute）的一员而骄傲。在菲律宾，他的团队提倡为满足政府机构、学校和中小企业的计算需求而使用开源软件。他热爱技术、驾车旅行并且每日都关注犹他爵士队的新闻。

致 谢

感谢 Kylie Johnston 的巨大耐心、组织工作以及在项目管理过程中的好脾气。

感谢我们优秀的文字编辑——Ami Knox 和 Nicole Flores。

感谢我们的制作编辑——Liz Berry。

感谢 Michelle Lowman 的引导和建议。

感谢 Frank Pohlmann 同意整件事情。

感谢 Donna Benjamin 的精美插图和资料。

感谢 Apress 的整个团队。

目 录

第一部分 人 门

第 1 章 Linux 介绍.....2	第 3 章 Linux 基础..... 49
1.1 Linux 发行版.....2	3.1 准备开始..... 49
1.1.1 Red Hat Enterprise Linux3	3.2 Linux 与 Microsoft Windows 51
1.1.2 CentOS.....4	3.2.1 GUI桌面..... 51
1.1.3 The Fedora Project.....4	3.2.2 命令行..... 52
1.1.4 Debian Linux4	3.3 远程访问..... 56
1.1.5 Ubuntu4	3.4 获得帮助..... 58
1.1.6 Gentoo.....5	3.5 用户与组..... 59
1.1.7 那么我们应该选择哪一种 发行版呢?5	3.6 服务与进程..... 60
1.1.8 本书涉及了哪些 发行版呢?6	3.7 软件包..... 61
1.2 选择硬件.....6	3.8 文件与文件系统..... 62
1.3 支持的硬件.....7	3.8.1 文件类型与权限..... 66
1.4 获取系统软件.....7	3.8.2 链接..... 71
1.5 获得支持.....8	3.8.3 用户、组与所有权..... 71
1.6 小结.....8	3.8.4 大小和空间..... 72
第 2 章 安装 Linux.....9	3.8.5 日期与时间..... 73
2.1 Live CD 与虚拟机.....10	3.9 文件操作..... 73
2.1.1 LiveCD.....10	3.9.1 读文件..... 74
2.1.2 虚拟机10	3.9.2 搜索文件..... 76
2.2 Red Hat Enterprise Linux 的 安装..... 11	3.9.3 复制文件..... 77
2.3 Ubuntu 的安装33	3.9.4 移动与重命名文件..... 79
2.4 故障检修.....47	3.9.5 删除文件..... 80
2.4.1 诊断信息47	3.9.6 链接文件..... 80
2.4.2 重新安装47	3.9.7 编辑文件..... 81
2.4.3 故障检修资源47	3.10 小结..... 83
2.5 小结.....48	第 4 章 用户和组..... 84
	4.1 登入时发生了什么事? 84
	4.2 用户与组操作..... 85
	4.2.1 sudo介绍 85

4.2.2 创建用户	86	6.1.1 从接口开始	139
4.2.3 创建组	89	6.1.2 在GUI下配置接口	142
4.2.4 删除用户和组	91	6.1.3 使用网络脚本配置 网络	151
4.2.5 通过GUI管理用户和组	92	6.1.4 添加路由与转发 数据包	161
4.2.6 密码	94	6.2 一般网络故障检修	165
4.2.7 密码时效	95	6.2.1 Ping!	166
4.2.8 停用用户	97	6.2.2 MTR	167
4.2.9 存储用户和组数据	97	6.2.3 TCP/IP 101	168
4.2.10 配置shell和环境	100	6.2.4 tcpdump命令	169
4.3 控制对主机的访问	102	6.2.5 Netcat工具	171
4.4 sudo 命令详解	106	6.2.6 dig它?	172
4.5 小结	111	6.2.7 其他故障诊断工具	174
第5章 启动与服务	112	6.3 Netfilter 与 iptables	175
5.1 当主机启动时发生了 什么事?	112	6.4 Netfilter/iptables 是如何 工作的?	175
5.1.1 BIOS	112	6.4.1 表	176
5.1.2 引导程序	113	6.4.2 链	176
5.1.3 操作系统	114	6.4.3 策略	177
5.2 了解 GRUB 引导程序	114	6.4.4 网络地址转换	178
5.2.1 配置GRUB	115	6.4.5 使用iptables命令	179
5.2.2 使用GRUB菜单	117	6.4.6 对Red Hat主机上默认 规则的解释	184
5.2.3 保护引导程序	118	6.5 配置范例网络	189
5.3 启动之后发生了什么?	118	6.5.1 我们的配置	189
5.3.1 配置init	119	6.5.2 其他防火墙配置 工具	209
5.3.2 在运行级别之间移动	120	6.6 TCP Wrappers	210
5.4 管理服务	121	6.7 小结	210
5.4.1 管理Red Hat上的 服务	123	第7章 软件包管理	212
5.4.2 管理Ubuntu上的服务	127	7.1 软件包管理介绍	212
5.4.3 Upstart: 一种新方式	130	7.2 Red Hat 上的软件包管理	214
5.5 关闭和重启主机	131	7.2.1 准备开始	214
5.6 使用定时任务调度服务与 命令	132	7.2.2 软件包更新器程序	215
5.7 小结	135	7.2.3 软件包管理器程序	220
第6章 网络与防火墙	136	7.2.4 Red Hat网络 (RHN)	222
6.1 网络与连网概论	136		

7.2.5 Yellowdog Updater
Modified (Yum)230

7.2.6 Red Hat软件包管理
(RPM)234

7.2.7 通过源代码创建RPM
软件包238

7.3 Ubuntu 上的软件包管理239

7.4 使用 Synaptic 进行包管理249

7.4.1 添加软件250

7.4.2 移除软件251

7.4.3 管理软件源252

7.4.4 更新与升级254

7.4.5 使用更新管理器255

7.5 使用 dpkg 包管理256

7.5.1 查看软件包细节258

7.5.2 安装软件包259

7.5.3 卸载软件包260

7.6 编译源代码.....260

7.6.1 配置262

7.6.2 编译和生成263

7.6.3 安装264

7.6.4 卸载265

7.7 小结..... 265

第 8 章 存储管理与灾难恢复..... 266

8.1 存储器基础..... 266

8.1.1 设备..... 266

8.1.2 分区..... 268

8.1.3 文件系统..... 274

8.2 使用文件系统..... 282

8.2.1 自动挂载..... 284

8.2.2 检查文件系统的利用率... 286

8.3 RAID..... 287

8.3.1 RAID的类型 288

8.3.2 创建阵列..... 290

8.4 逻辑卷管理..... 296

8.4.1 创建卷与卷组..... 297

8.4.2 扩充逻辑卷..... 299

8.4.3 缩减逻辑卷..... 300

8.4.4 用GUI管理LVM..... 301

8.5 故障恢复..... 304

8.5.1 引导程序的问题..... 307

8.5.2 磁盘故障..... 308

8.6 小结..... 310

第二部分 让 Linux 为你工作

第 9 章 基础架构服务: NTP、DNS、
DHCP 和 SSH312

9.1 网络时间协议312

9.2 域名系统.....316

9.2.1 根服务器316

9.2.2 查询域名服务器318

9.2.3 运行缓存DNS322

9.2.4 权威DNS服务器326

9.2.5 动态DNS337

9.3 动态主机配置协议337

9.3.1 安装与配置337

9.3.2 静态租约分配339

9.3.3 动态DNS更新341

9.3.4 手动修改DNS输
入项..... 347

9.4 Secure Shell..... 348

9.4.1 创建和分配密钥..... 348

9.4.2 使用SSH代理 349

9.4.3 调整SSH配置 350

9.4.4 执行快速又安全的
文件传输任务..... 353

9.5 小结..... 354

第 10 章 邮件服务 355

10.1 电子邮件是如何工作的..... 355

10.1.1 发送一封电子邮件时
发生了什么事? 356

10.1.2 电子邮件发送之后 发生了什么事?	358	11.3 安装网络站点	437
10.2 配置电子邮件	359	11.3.1 网络交流	438
10.2.1 安装	360	11.3.2 webmail	443
10.2.2 启动Postfix	362	11.3.3 配置SquirrelMail	445
10.2.3 了解Postfix配置	363	11.3.4 其他Web应用程序	448
10.2.4 初始配置	365	11.4 Squid Cache	448
10.2.5 测试Postfix	366	11.4.1 配置	449
10.2.6 选择邮箱格式	369	11.4.2 客户端配置	451
10.3 Postfix 扩展配置	372	11.4.3 透明性	451
10.3.1 使用加密功能	372	11.5 小结	452
10.3.2 身份验证	382	第 12 章 文件和打印共享	453
10.4 获得 Postfix 相关的帮助	390	12.1 使用 Samba 和 NFS 共享 文件	453
10.5 防止病毒和垃圾邮件	391	12.2 Samba	454
10.5.1 与垃圾邮件作战	391	12.2.1 给Samba添加用户	463
10.5.2 防病毒	401	12.2.2 向域中添加主机	464
10.6 配置 IMAP 和 POP3	407	12.2.3 Samba所需的IP表规则	466
10.6.1 IMAP	407	12.2.4 在Linux上挂载Samba的 共享目录	467
10.6.2 POP3	407	12.2.5 使用system-config-samba 图形用户界面	467
10.6.3 二者有什么区别?	407	12.2.6 资源	471
10.6.4 在IMAP和POP3之间 选择	408	12.3 NFS 共享文件: Linux 到 Linux	471
10.6.5 Dovecot介绍	408	12.4 管理文档	473
10.7 虚拟域与虚拟用户	414	12.4.1 使用文档管理系统	473
10.8 小结	415	12.4.2 开源DMS Knowledge Tree	473
第 11 章 Web 服务和 SQL 服务	416	12.4.3 安装KnowledgeTree	474
11.1 Apache 网页服务器	416	12.4.4 管理KnowledgeTree	478
11.1.1 安装和配置	416	12.4.5 处理文档	481
11.1.2 访问控制	426	12.4.6 启动和停止Knowledge Tree文档管理系统	486
11.1.3 模块	427	12.4.7 带SSL的安全 KnowledgeTree	486
11.1.4 文件和目录许可	428	12.4.8 资源	487
11.2 MySQL 数据库	429	12.5 打印服务	487
11.2.1 安装	429		
11.2.2 测试服务器	431		
11.2.3 基本的InnoDB优化 技术	432		
11.2.4 基本的MySQL管理	435		

12.5.1	CUPS	487	15.2.3	安装Zimbra	566
12.5.2	Samba和打印服务: 给桌面系统添加打印机.....	494	15.2.4	Zimbra安装后续的 配置菜单	568
12.6	小结	495	15.2.5	更改防火墙	572
第 13 章	备份和恢复	496	15.2.6	Zimbra管理控制台	573
13.1	灾难恢复计划	496	15.2.7	创建Class of Service.....	574
13.2	备份过程	497	15.2.8	添加新用户	579
13.3	网络备份	499	15.2.9	邮箱别名和邮件分发 列表	584
13.4	使用 rsync	500	15.2.10	添加资源	587
13.5	使用 Bacula	509	15.2.11	添加Zimlet	588
13.5.1	获取软件	510	15.2.12	添加一个SSL认证	590
13.5.2	配置Bacula	514	15.2.13	全局设置	594
13.5.3	使用bconsole 管理Bacula	525	15.2.14	监控Zimbra	597
13.5.4	使用Bacula备份 数据库	528	15.3	使用 Zimbra	598
13.5.5	介绍Bat控制台	531	15.3.1	使用电子邮箱	599
13.6	小结	536	15.3.2	使用Zimlet	601
第 14 章	构建 VPN 网络	537	15.3.3	共享文件夹、地址簿、 文档及其他资源	603
14.1	示例网络	537	15.4	迁移已有的邮件服务	609
14.2	OpenVPN 介绍	538	15.5	小结	610
14.2.1	安装OpenVPN	539	第 16 章	目录服务	611
14.2.2	启动和停止OpenVPN	539	16.1	什么是 LDAP	611
14.2.3	配置OpenVPN	540	16.2	总则	614
14.2.4	用OpenVPN发布 总公司资源	550	16.3	部署	616
14.2.5	为移动用户创建 VPN连接	552	16.4	安装	617
14.3	OpenVPN 故障排除	559	16.4.1	Red Hat安装指导	617
14.4	小结	559	16.4.2	Ubuntu安装指导	617
第 15 章	协作服务	560	16.5	配置	618
15.1	Zimbra	561	16.5.1	创建模式	623
15.2	安装 Zimbra	563	16.5.2	访问控制列表	625
15.2.1	安装前提	563	16.5.3	启动slapd守护进程	629
15.2.2	下载和主机前期准备	564	16.5.4	设置LDAP客户端	631
			16.6	LDAP 管理及其工具	631
			16.6.1	LDIF文件与添加用户	632
			16.6.2	使用LDIF文件添加 用户	633

16.6.3 搜索LDAP树.....636	第 18 章 日志记录与监控.....699
16.6.4 从LDAP中删除条目.....637	18.1 日志记录..... 699
16.6.5 密码策略覆盖.....638	18.1.1 配置syslog..... 700
16.6.6 测试访问控制列表.....639	18.1.2 启动与配置syslog 守护进程..... 705
16.6.7 备份LDAP目录.....640	18.1.3 使用logger工具测试 日志记录..... 707
16.6.8 LDAP账户管理: 基于Web的GUI.....642	18.1.4 日志管理与轮替..... 707
16.6.9 安装与配置.....642	18.2 日志分析与关联..... 710
16.6.10 为LAM添加Apache 虚拟主机.....643	18.2.1 SEC简介..... 710
16.7 与其他服务整合.....649	18.2.2 安装SEC..... 712
16.7.1 单点登录: 集中 Linux认证.....649	18.2.3 运行SEC..... 712
16.7.2 PAM运行机制.....655	18.2.4 使用SEC..... 715
16.7.3 LDAP和Apache认证.....658	18.2.5 SEC排错..... 723
16.7.4 LDAP与知识树 DMS整合.....661	18.3 监控..... 723
16.8 小结.....665	18.3.1 Nagios简介..... 723
第 17 章 性能监控与优化..... 666	18.3.2 安装Nagios..... 725
17.1 基本的健康状况检查.....666	18.3.3 启动Nagios..... 726
17.1.1 CPU利用率.....666	18.3.4 Nagios配置..... 726
17.1.2 内存利用率.....667	18.3.5 搭建Nagios控制台..... 742
17.1.3 磁盘空间.....668	18.3.6 Nagios疑难解答..... 748
17.1.4 日志.....668	18.4 小结..... 748
17.2 高级工具.....669	第 19 章 配置管理.....749
17.2.1 CPU和内存利用率.....669	19.1 自动配置..... 749
17.2.2 交换空间的使用.....676	19.1.1 在Red Hat中使用 Cobbler自动化配置..... 750
17.2.3 磁盘存取.....677	19.1.2 在Ubuntu中进行自动化 配置..... 759
17.3 持续性能监控.....678	19.1.3 Kickstart和Preseed..... 767
17.3.1 SNMP.....678	19.2 配置管理..... 776
17.3.2 Cacti.....681	19.2.1 Puppet简介..... 776
17.4 性能优化.....692	19.2.2 安装Puppet..... 778
17.4.1 资源限制.....693	19.2.3 配置Puppet..... 778
17.4.2 sysctl和proc文件系统.....695	19.2.4 连接第一个客户端..... 780
17.4.3 存储设备.....696	19.2.5 创建第一个配置..... 782
17.4.4 文件系统调整.....696	19.2.6 应用第一个配置..... 784
17.5 小结.....697	

19.2.7 为多个主机定义配置	785	20.3.2 创建Xen虚拟机	803
19.2.8 相关资源	788	20.3.3 管理Xen虚拟机	806
19.2.9 使用模板	789	20.3.4 自动启动Xen虚拟机	810
19.2.10 定义	791	20.4 使用 KVM 安装虚拟机	810
19.2.11 更多 Puppet	792	20.4.1 为KVM虚拟化准备	
19.2.12 Puppet排错	793	服务器: 网络方面	811
19.3 小结	794	20.4.2 在Ubuntu服务器上	
第 20 章 虚拟化	795	设置KVM	811
20.1 虚拟化解决方案	795	20.4.3 在KVM上安装一个	
20.1.1 VirtualBox	795	Windows客户端	
20.1.2 VMware	796	操作系统	812
20.1.3 Xen	796	20.4.4 在KVM上安装一个	
20.1.4 KVM	798	Ubuntu服务器客户端	
20.1.5 OpenVZ	798	操作系统	813
20.2 使用 VirtualBox	798	20.4.5 使用虚拟机管理	
20.2.1 安装VirtualBox	798	器管理KVM虚拟机	813
20.2.2 使用VirtualBox创建		20.5 使用 OpenVZ 的虚拟化方案	816
虚拟机	799	20.5.1 安装	816
20.3 使用 Xen 安装虚拟机	802	20.5.2 创建OpenVZ虚拟机	818
20.3.1 为使用Xen准备计算机		20.5.3 OpenVZ虚拟机基本	
环境	802	管理	820
		20.6 小结	820

第一部分

入门

第 1 章 Linux 介绍



James Turnbull, Peter Lieverdink, Dennis Matotek 编写

已经决定在免费开源 (FOSS) 平台上开展工作了吗？那恭喜并欢迎你来到 Linux 开源软件的世界。本章将带你走出部署这样一个平台的第一步，其中包括如何挑选平台或者叫发行版，准备系统所支持的合适的硬件，以及找到所需的软件等内容。本章还提供了一些可以帮助支撑 Linux 环境的资源地址。在接下来的第 2 章，我们再介绍如何安装第一台 Linux 主机。

1.1 Linux 发行版

什么是 Linux 发行版呢？简单来讲，发行版就是在 Linux 内核之上运行的应用程序、软件包、管理工具以及其他功能部件的集合。内核是所有发行版所共有的部分（有时发行版的维护者会将内核进行个性化改动），同时，所有发行版的核心部分运行的都是 Linux。

■注：你可能会问，什么是内核呢？别慌，你会弄明白的。内核就是所有计算机操作系统的核心，它通常位于允许操作系统和计算机硬件进行交互的结构层次。内核提供了支持运行硬盘驱动器、网卡、内存 (RAM) 以及其他硬件组件的软件。在 Linux 世界里，内核是基于 Linux 的奠基人、开发完成者 Linus Torvalds 的开发源代码的。目前，一个由开源开发者组成的团体正在对内核进行维护，同时按照一个完整的软件生命周期的过程对内核进行更新。所有发行版中都拥有这样一个某个版本的内核，而且，就像 Windows 和其他操作系统一样，这个内核也可以通过更新、升级来获得新的特征或进行漏洞修复。

乍看起来 Linux 发行版的世界有点使人困惑。你可能会想：“如果都是 Linux，它们为什么还要有那么多不同的名字呢，我又应该选择哪一个呢？”大家都可能听说过如 Red Hat、Fedora、Debian 以及比较古怪的 Ubuntu (Ubuntu 是祖鲁语，大致可以译为“与人为善”) 等名称。本章中，我们将会介绍什么是发行版，并描述它们之间是如何的不同，同时在选择正确发行版的策略方面提供有益的建议。

各种发行版在很多方面存在着差异，其中最重要的三点如下所示。

- 发行的目的。
- 结构与组装方式。
- 所支持的模块。

首先，不同的 Linux 发行版通常是不同的使用目的所设计的，同时也提供了不同的用户体验。有些发行版用来作服务器，有的用来作桌面应用，而有些，如在嵌入式系统中的，则是用来完成特殊的应用。Linux 的主要应用还是倾向于作为服务器使用。尽管有越来越多的 Linux 作为桌面应用出现，但还是没能挑战 Windows 和 Apple OS X 的桌面应用市场。

不同 Linux 发行版间的第二个主要的区别在于它们的结构不同。有些发行版把结构设置和文件放在同一个路径下，而其他的却全然不同。另外，不同的发行版间安装和升级应用程序（一般是通过软件包进行安装）的过程也不尽一致。很多发行版都使用不同的应用程序安装和管理工具（一般称为软件包管理工具）。如果经常工作于不同的发行版之间，这些不同可能使人困惑，同时也会使管理工作困难重重。在第 19 章，我们将讨论配置管理工具以及如何解决诸如此类的问题。

第三个不同是，各种发行版的维护、支持模式各有不同。有的，如 Debian、CentOS 和 Fedora 等，都是由志愿团体维护的。其他的，如 Red Hat Enterprise Linux 和 Ubuntu，是由商业团体维护和支持的。尽管其中的软件还是开源的，但是你还是得付费获得支持和维护。很多 Linux 商业团体通过销售维护和支持服务来支撑其自身发展。

让我们来看看一些可用的选择吧，尽管这并不是一个全面的列表，但其中囊括了大多数主流的发行版，同时也列出了选择各种特定平台的一些理由。我们还会对相似的发行版进行分类，并特别关注从两个主要发行版中衍生出的版本，这两个主要的发行版分别是 Red Hat 和 Ubuntu（Ubuntu 自身也是 Debian 发行版的衍生版）。

■注：一个发行版是如何从其他的发行版中衍生出来的呢？是这样的，开源软件就意味着源代码对所有的开发者都是开放的，每一位开发者都可以从一个发行版中选取他们想要的部分特性，从而开发出自己的发行版。众多发行版的出现正是由于有一个或一群开发人员决定开发基于其他发行版的自己的版本。这些新的衍生版本一般都有自己的品牌名和特性。有的和自己的父版本很相似，有的就自循其路了。

1.1.1 Red Hat Enterprise Linux

Red Hat Enterprise Linux (<http://www.redhat.com/rhel/>) 是一个流行的商业支持的 Linux 平台。它有很多版本，最常见的两个是 Red Hat Enterprise Linux（又叫 RHEL）和 Red Hat Enterprise Linux Advanced Platform（RHELAP）。这两个版本的主要区别在于它们所支持 CPU 数量的不同，RHEL 最多支持两个 CPU，而 RHELAP 却支持无限个。

鉴于发行商专注的支持和有效的服务等级，Red Hat 平台通常被企业组织用来作为服务器平台。Red Hat，以及其他很多基于 Red Hat 的发行版，都使用 Red Hat Package Management（RPM）软件包管理系统。

写作本书时，RHEL 的费用从 350 美元一年的基础支持到 1300 美元一年的高级支持不等。其高级姊妹篇，RHELAP，一年的费用，根据所提供的支持服务等级的不同从 1500 美元到 2500 美元不等。这笔费用会给用户带来所使用的发行版的技术支持和所有必要的补丁和更新。

Red Hat 以前也是由一个志愿团体维护的，但是随着这个发行版对商业团体的技术基础变得越来越重要，就有人乐于付费以取得有保障的支持服务了。起初的志愿团体也仍然存在，比如 Fedora 项目开发团体。

1.1.2 CentOS

CentOS(<http://www.centos.org>)是 Red Hat Enterprise Linux 平台的衍生版。尽管它们基于相同的源代码，但是 CentOS 是免费的（没有 Red Hat 的支持）。想使用 Red Hat 平台及其稳定的性能而又不想为额外服务付费的话，CentOS 是最好的选择。它使用了与 RHEL 一样的系统组织模式，即 RPM，也有很多与 Red Hat 产品相同的管理工具。

1.1.3 The Fedora Project

The Fedora Project (<http://fedoraproject.org>)是由 Red Hat 和社区共同维护的发行版。它是 Red Hat Enterprise Linux 的一个衍生版本，也为发布商业产品提供了一个预开发平台。由于是 Red Hat 发起的，Fedora 就成了 Red Hat 很多新特性的实验场。因此，有人认为把 Fedora 应用于商业领域有些太激进。引入到 Fedora 的很多特性通常会出现在新的 RHEL 发行版中，Fedora 也使用了 RPM 软件包管理以及很多和 RHEL 中相同的管理工具。

1.1.4 Debian Linux

Debian Linux 发行版 (<http://www.debian.org>)是一个由社区开发和管理的免费版本，它拥有许多活跃的开发者和用户。Debian 始于 1993 年，根据一个社会契约创建 (http://www.debian.org/social_contract)。Debian 发行版致力于自由、开放，同时坚持专注于提供用户所需平台。

1.1.5 Ubuntu

由南非技术专家、企业家马克·沙特沃斯 (Mark Shuttleworth) 发起的 Ubuntu (<http://www.ubuntu.com/>)是一个免费的、基于 Debian Linux 平台的操作系统。它由社区开发，以 6 个月为周期进行升级。Ubuntu 也拥有来自它的合作组织 Canonical 的商业援助，同时也有第三方团体提供支持。Ubuntu 在作为桌面应用和服务器应用时表现出不同的特性。很多评论家相信，Ubuntu 处处显示出来的生命力和稳定性昭示着 Linux 作为桌面平台应用的不断增长。很多人认为 Ubuntu 是最易于使用 and 理解的 Linux 平台中的一个，它的开发者也将易用性和良好的用户体验作为它的定位标准。Ubuntu 使用 Debian 的软件包系统和其中一部分管理工具。

1.1.6 Gentoo

Gentoo 发行版 (<http://www.gentoo.org/>) 是另一个由社区开发的平台。这个版本值得注意的原因是它提供了可以根据你的硬件编译整个发行版的选项。这就使得我们可以自定义每一个选项以适应特定的硬件环境，但是花费的时间可能会比较长。对于技术不足又不想重新编译的用户，Gentoo 也可以以预编译的形式进行安装。Gentoo 还以其经常用作 MythTV 平台而众所周知，此平台是一种类似于 Microsoft Media Center 的开源的媒体中心应用程序。Gentoo 使用一种独特的称作 Portage 的软件包管理系统。

■提示：你可以在 DistroWatch (<http://distrowatch.com>) 了解到在 Linux 世界中存在的各种发行版信息。

1.1.7 那么我们应该选择哪一种发行版呢？

挑选一个特定的发行版应该基于公司的预算、技术以及需求。当然，我们还是推荐大家选择 Red Hat 的衍生版、Ubuntu（一种基于 Debian 的发行版）或 Debian。上述版本都得到了其维护者组织或社区的良好支持。

■注：你可以在 <http://www.zegeniastudios.net/ldc/index.php> 找到关于如何选择一个合适、可用的 Linux 发行版的在线自动问答系统，也可以在 http://wiki.linuxquestions.org/wiki/choosing_a_Linux_distribution 找到相关的资料。

除了 Red Hat Enterprise Linux 需要一个付费的支持协议才能得到升级和补丁外，其他所有我们讨论过的发行版都是免费的。我们可以对其进行下载、安装而不需要支付任何许可费用。

■注：大家也可以免费获得 Red Hat Enterprise Linux 并进行安装而不必支付许可费用——唯一的麻烦是没有支持协议，系统将不能获得任何升级，而这可能会留给我们一个充满漏洞的不安全的主机。

以上讨论过的发行版中有几款可以提供商业支持，如 Red Hat Enterprise Linux 或 Ubuntu（拥有它的合作伙伴 Canonical 提供的支持）。如果你的技术并不是很强，这样的发行版是值得考虑的。大家也应该知道当地支持者也可能会提供技术方面的支持。比如，有些 IT 公司及系统集成商就提供 Linux 支持，而且经常有些从事 IT 支持业务的中小型公司也可以提供相关的支持服务。

■提示：你可以在 <http://webapps.ubuntu.com/marketplace/> 找到通过 Ubuntu 市场提供 Ubuntu Linux 支持的当地服务提供商。

如果不想为第三方或销售商提供的商业技术支持买单的话，或许可以考虑从以其庞大的

活跃社区著称的一系列发行版中挑选一个，在这些社区中我们可以得到支持与帮助。尤其是近几年由于更多新的使用者对这个发行版的熟知，Ubuntu 的支持资源得到增长。

最后，不要忽略自己的亲身体验。亲自体验一下各种发行版。尝试一下 LiveCD，安装几个发行版，还要感觉一下各种不同的管理工具和界面。永远都不要轻视自己对发行版的适用性以及在其上工作的舒适性的感觉。

1.1.8 本书涉及了哪些发行版呢？

正如上述提到的一样，两个主流的选择一个是 Red Hat 或其衍生版（如 CentOS 和 Fedora），另一个是 Ubuntu 或与其相关的发行版。于是我们决定在本书中主要介绍两类平台，一个是 Red Hat 衍生的发行版，另一个是 Debian 衍生的发行版。我们选择这两类平台是因为它们很好地代表了发行版中的两个主要的家族。同时也方便我们介绍主要的配置选项和配置风格，软件包管理工具以及众多现有 Linux 发行版所使用的相关管理技巧。

本书还详尽地介绍了在以下平台上部署应用程序和工具所必须的资料。

Red Hat Enterprise Linux 或像 CentOS 和 Fedora 等基于 Red Hat 的发行版。

Ubuntu 或其他基于 Debian 的发行版。

在提供特定的实例时，我们选择使用 Red Hat Enterprise Linux 5 和 Ubuntu LTS Server 8.04 版进行说明。

■注：LTS 是“long term support(长期支持)”的缩写。Ubuntu 项目组每 6 个月升级一次它的服务器和桌面发行版。Ubuntu 项目组保证每一个 LTS 发行版在其发行后会对其进行为期 5 年的支持，包括修复漏洞以及为安全问题打补丁。

我们在每一章都会为每种发行版的配置提供一些实例并指出各发行版间的不同，如配置文件的位置以及软件包的名称等。

1.2 选择硬件

对于如何选择合适硬件的详细分析已经超出了本书所论述的范围。我们一般会建议大家购买具有足够可信度和良好技术支持的硬件来满足公司的需求。如果平台需要具有足够的稳定性和高可用性，那么就应该购买具有冗余特性的硬件，比如具有后备电源供应等。还应该适当地购买一些支持能力，如备件以及网站、电话或在线的支持等。

■注：另一个选择是从如 Rackspace (<http://www.rackspace.com>) 或 Linode(<http://www.linode.com>) 等服务提供商处购买专用的或虚拟的服务器。类似这样的公司在 Internet 上提供各种发行版和各种配置的 Linux 服务器主机。通常需要按月或年支付租金。如此一来我们就可以远程连接服务器并且配置它。有些公司还为不同的使用目的提供安装好的或配置好的主机。我们会在第 20 章中看到使用网络主机和虚拟服务的内容。

1.3 支持的硬件

除了购买正确的硬件外，还应该重视一些重要的选项以及做一些性能方面的考虑。最重要的一个考虑是并不是所有的硬件都被 Linux 操作系统所支持。尽管这样的情况很少见，但是还是有一些硬件组件（例如，一些无线网卡）在一部分甚至所有 Linux 发行版中没有驱动程序从而是不被支持的。

应该确定所购买的硬件是否被所选择的发行版支持。绝大多数发行版都有硬件支持列表（HCL），可以利用这些列表来确认系统对所购硬件的支持与否。以下是最近被维护过的 HCL 站点。

- <https://hardware.redhat.com/>（Red Hat, CentOS 和 Fedora 相关）。
- <https://wiki.ubuntu.com/hardware suport/> (Ubuntu)。
- <http://kmuto.jp/debian/hcl/wiki/>（Debian，也是 Ubuntu 相关的）。
- <http://www.linuxquestions.org/hcl/index.php>（通用列表）。

也有一些大规模硬件厂商会提供拥有 OEM Linux 软件的系统。我们也可以从诸如 Dell、HP 以及 IBM 这些提供硬件可以支持的 Linux 发行版列表的厂商那里选择硬件。

■注：在后续的章节里当遇到特定的应用程序和工具时，我们会讨论各种各样详细的性能话题。

1.4 获取系统软件

安装第一个主机应该从哪里开始呢？首先，要获取所需要的系统软件的副本。获得基本的操作系统软件有很多种方式。有的发行版出售 CD-ROM 和 DVD，其他的提供可下载的 ISO 镜像文件（也有的同时通过两种方式提供）。也有的发行版提供通过网络或 Internet 在线安装的方式。

■注：我们将会在第 19 章看到自动的网络提供服务器的安装过程。

以下是可以获取 CD-ROM 和 DVD 光盘的站点列表。

- <http://www.ubuntu.com/GetUbuntu/>
- <http://www.debian.org/distrib/>
- <http://www.centos.org/modules/tinycontent/index.php?id=15>
- <http://www.gentoo.org/main/en/mirrors2.xml>
- <https://www.redhat.com/apps/download/>
- <http://fedoraproject.org/get-fedora/>

下载到所需的系统软件后，可以把 ISO 镜像刻录成 CD 或 DVD。下面的 URL 地址指向

的文档描述了如何将一个 ISO 文件刻录成 CD 或 DVD 光盘。

- <http://pcsupport.about.com/od/toolsoftthetrade/ht/burnisofile.htm>
- http://www.petri.co.il/how_to_write_iso_files_to_cd.htm
- <https://help.ubuntu.com/community/BurningIsoHowto>

或者如果你已经准备好了安装媒体，那就可以到第 2 章开始安装了。

1.5 获得支持

根据所选的 Linux 发行版的不同，获取帮助和支持的方式也千差万别。如果选择的是一个商业发行版，就可以联系销售商以获取所需的支持。对于非商业发行版，则可以在所选发行版的网站上发表问题帖子或者浏览相关文档。

另外，永远不要低估通过搜索引擎找到问题解决方案的力量。在全世界范围内使用 Linux 的许多人可能都碰到过和你同样的问题，而且发表过或写过相关的解决办法。

对于一些具体的发行版，下列站点将是非常有用的。

- Red Hat: <https://www.redhat.com/apps/support/>
- CentOS: http://bugs.centos.org/main_page.php
- Fedora: <http://fedoraproject.org/en/get-help>
- Debian: <http://www.debian.org/support>
- Ubuntu: <http://www.ubuntu.com/support>
- Gentoo: <http://www.gentoo.org/main/en/support.xml>

查看其他发行版的站点可以了解到它们的支持机制。其他有用的站点如下所示。

- LinuxQuestions.org: <http://www.linuxquestions.org>
- LinuxHelp: <http://www.linuxhelp.com/>
- Linuxselfhelp: <http://www.linuxselfhelp.com/>
- ReallyLinux: <http://www.reallylinux.com/>

1.6 小 结

本章介绍了各种 Linux，包括本书将集中介绍的两个发行版：

- Red Hat Enterprise Linux
- Ubuntu

我们还讨论了选择一个特定发行版的理由，如何挑选合适的硬件以及哪里可以获得已选发行版的基本支持。在下一章中，我们将介绍如何安装本书中介绍的两个发行版。

第 2 章 安装 Linux



本章将介绍使用 Red Hat 企业版 Linux（Red Hat Enterprise Linux, RHEL）和 Ubuntu 服务器版本安装主机的整个过程。通过使用图形化安装工具，我们将展示每个发行版的安装过程，并详述安装过程中的选项。我们除执行基本的安装外，还会安装一个基本的网络、邮件和 DNS 服务器运行所需的软件包。如果此时不知道这些功能也不用担心——第 11 章、第 10 章和第 9 章将分别阐述网络、邮件和 DNS。

■提示：我们推荐完整地阅读本章，包括那些涉及 Red Hat 和 Ubuntu 安装过程的章节，以便获得对 Linux 主机安装最好的理解。

本章将从“Red Hat Enterprise Linux 的安装”一节中安装一个基于 Red Hat 的发行版开始。虽然该节中的屏幕截图是针对 RHEL 的，但 CentOS 和 Fedora 的安装过程来源于 RHEL，工作方式很类似。因此如果已经选择了这些发行版中的任何一个，都应该能从讲解中轻松识别出这些发行版的安装过程。你会发现，这对于 Red Hat 衍生的发行版的大部分配置和管理都是正确的。

如果选择了 Ubuntu，则可在“Ubuntu 服务器的安装”一节中找到 Ubuntu 安装的完整介绍。Ubuntu 来源于 Debian，但它们的安装过程不一样。然而其配置和选项很相似，通过了解 Ubuntu 的安装过程，你应该能够识别 Debian 或者 Debian 衍生的其他发行版的安装过程。

■注：如果你想使用基于 CD/DVD 的安装方法以及它所提供的图形化安装程序，那么就需要主机配备监视器和键盘，有鼠标则更好。使用这些外围设备，你可以有效地与安装工具交互。第 19 章将说明如何完成无人看管或者无头（没有监视器）的安装。

在第 7 章介绍 Linux 上的软件安装以及在第 19 章分析自动安装和编译的方法时，我们还将详述可能的安装选项。

■注：发行版不同，安装屏幕和选项也随之不同。如果本章给出的屏幕截图和你安装过程中的截图不完全符合，请不要恐慌。一般地，大部分安装选项与安装步骤在不同发行版之间还是相似的。

2.1 LiveCD 与虚拟机

在进行第一次安装之前，为了解 Linux，我们需要探讨另外两类选项：LiveCD 和虚拟机。在构建实体服务器之前，尝试了解这些内容也许是有用的。利用这些方式可以探究一个 Linux 发行版，还可以看看如何以最少的时间和基础设施投资来使用它。

2.1.1 LiveCD

LiveCD 是可在计算机上从 CD 或者 DVD 运行的发行版。LiveCD 无需在计算机上安装任何软件就可把自己装载到内存。这就是说，你可以试用某个发行版，然后移除 CD，重新启动就又回到已有的操作系统，这样无需改变计算机上的任何东西，就可轻松浏览和测试 Linux 发行版。在 http://en.wikipedia.org/wiki/Live_CD 上可以找到 LiveCD 的更多信息。

你可以查找流行发行版的 LiveCD，如下面所示。

- Ubuntu: <https://help.ubuntu.com/community/LiveCD>
- Fedora: <http://fedoraproject.org/wiki/FedoraLiveCD>
- Debian: <http://debian-live.alioth.debian.org/>

在 <http://www.livecdlist.com/> 上还可以找到众多可用 LiveCD 的完整列表。

2.1.2 虚拟机

Linux 发行版还可以在虚拟机上运行。虚拟机是主机的软件实现，运行起来就像真实的主机。多个虚拟的主机可以运行在一台实体主机上。虚拟化应用程序和服务器的实例有 VMware (<http://www.vmware.com/>)、VirtualBox (<http://www.virtualbox.org/>)，还有开源的备选方案，比如其中的 Xen (<http://www.xen.org/>)。也可以从托管公司购买虚拟主机。

■注：本章将演示如何安装 Linux 主机。这些操作指南详述“裸机”而不是虚拟主机的安装步骤。裸机安装与虚拟机安装之间的差别相对较小。对于虚拟主机来说，其中一个区别在于它可从 ISO 镜像安装，而不必先把 ISO 镜像烧录成 CD/DVD，再加载到 CD/DVD 驱动器。此外，虚拟主机安装使得构建和重建主机更容易，并且可以执行如创建不同种类主机的时间点备份这样的功能。

你可能还希望利用预先制作的虚拟工具，它们是借助虚拟化软件加载的 Linux 发行版的虚拟镜像。这些虚拟工具已经安装和配置，而且通常是为特定用途而创建的，如 VoIP 服务器、文件服务器或者邮件服务器。可在下列站点查看可用的虚拟工具列表。

- <http://www.vmware.com/appliances/>: VMware 下的虚拟工具。

- <http://virtualappliances.net/>: 多种虚拟化引擎下的 Ubuntu 虚拟工具。
- <http://jailtime.org/>: Xen 下的虚拟工具。

■注：第 20 章将更详细地探讨 Linux 虚拟机。

2.2 Red Hat Enterprise Linux 的安装

让我们从安装 Red Hat Enterprise Linux 主机开始。这里做以下几个假定。

- 所使用的 Red Hat Enterprise Linux ISO 来自 Red Hat 网站 (<https://www.redhat.com/apps/download/>), 并已经把它烧录到 CD 上。
- 所构建的只是一个基本的邮件、DNS 和网络服务器。
- 安装所用的服务器是新的, 先前没有安装任何操作系统。

首先, 把安装介质 (通常为 CD 或 DVD) 放进主机, 然后开机。

■注：如果正在构建虚拟机, 应该改为从原始 ISO 开始构建。虚拟机通常包括一个“虚拟 DVD”, 通过它挂载安装程序 ISO 来启动。

加载安装介质并启动主机之后, 将看到图 2-1 所示的 Red Hat 安装闪屏画面。



图 2-1 Red Hat Enterprise Linux 的闪屏画面

在闪屏画面中, 可以通过图形界面或者基于文本的安装机制启动安装程序。对于初次学习, 我们打算使用图形界面。

通过功能键可使用附加选项。F1 键回到主菜单。F2 键显示安装程序的附加启动选项，包括安装介质测试、内存检查以及附加磁盘驱动器的添加，这些选项可传递给 Red Hat。还可以让 Red Hat 启动基于网络的安装，这将在第 19 章讨论。

F3 键显示一般的帮助信息，并描述一些在安装 Red Hat 出现问题时可以使用的选项。F4 键描述一些可传递给内核用以定制安装的选项。

最后，F5 功能键显示挽救模式的选项。挽救模式假定已经加载 Linux，允许启动并修复或挽救已损坏的 Linux 安装。它会启动进入挽救模式的提示符，并允许挂载磁盘，编辑配置文件，以及访问其他有用的实用工具。在 http://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/Installation_Guide-en-US/s1-rescuemode-boot.html 上可以找到更多与挽救模式相关的信息。

不过，目前只需按回车键进入安装的下一阶段。下一画面提示检查安装介质。这次检查会扫描 CD 或 DVD 上的错误。如果愿意的话，这一过程可以跳过，然后继续下面的安装。

在图 2-2 中，安装主机的“anaconda”安装程序进程已经启动。anaconda 程序是安装 RHEL 的软件，它运行在 X 窗口系统——也就是大家所知的 X——此外它还有一个命令行模式。X 是 Linux 上常用的图形用户界面，第 3 章将会对它做更多的介绍。首先看到的是版本说明，然后可以单击 Next 到下一画面。



图 2-2 Red Hat Enterprise Linux 图形化安装程序

■提示：版本说明告诉你该版本和上一版本相比做了哪些改动。如果要升级主机，阅读并理解版本说明中改动的含义是一个好主意。

接下来的几个画面将选择主机的基本需求，如所用语言和键盘布局。图 2-3 中选择了中国大陆所使用的语言；你应该选择与自己相关的语言。

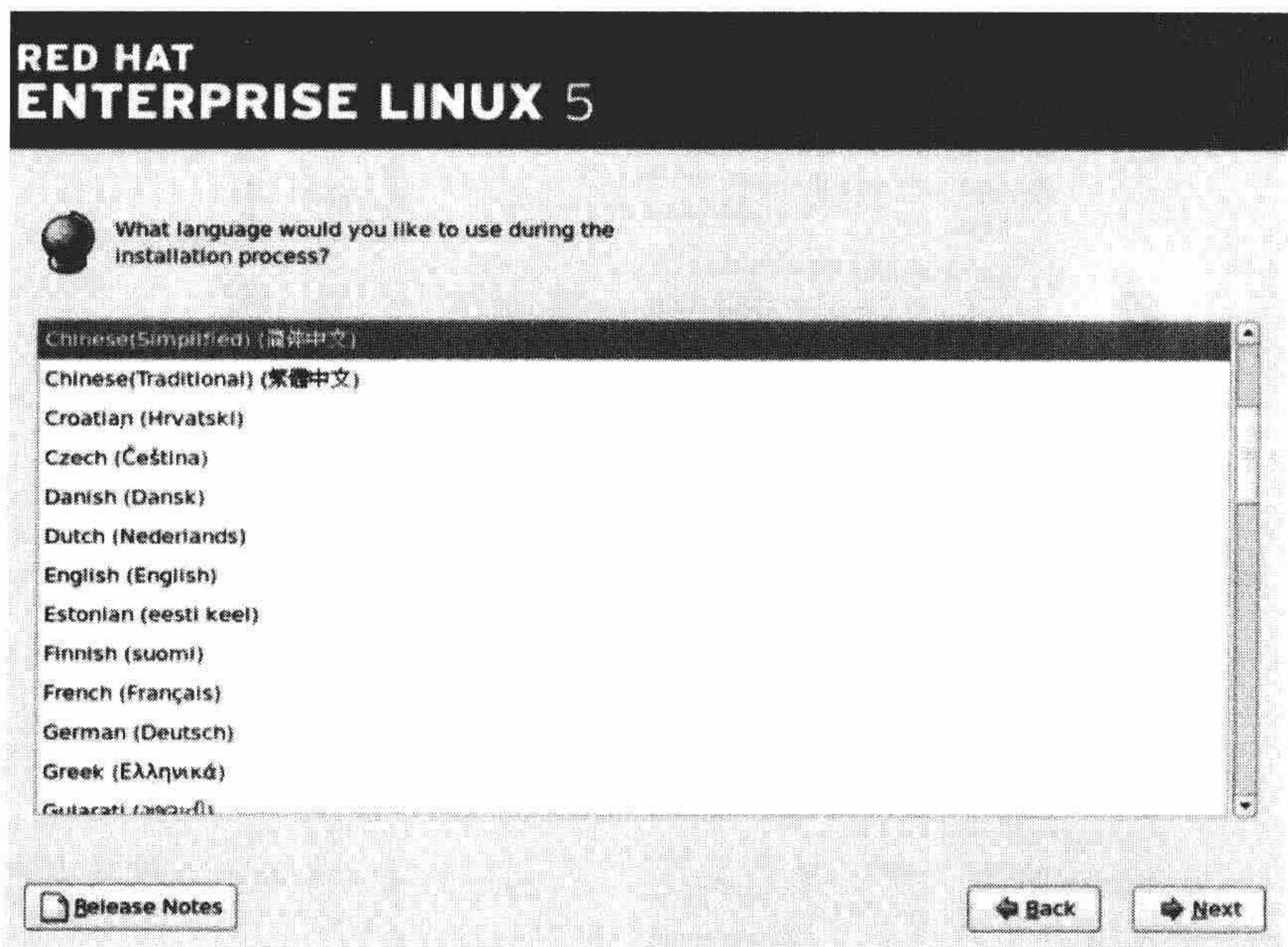


图 2-3 安装期间的语言选择

键盘映射影响键盘键位的默认布局。可如图 2-4 所示选择键盘布局。

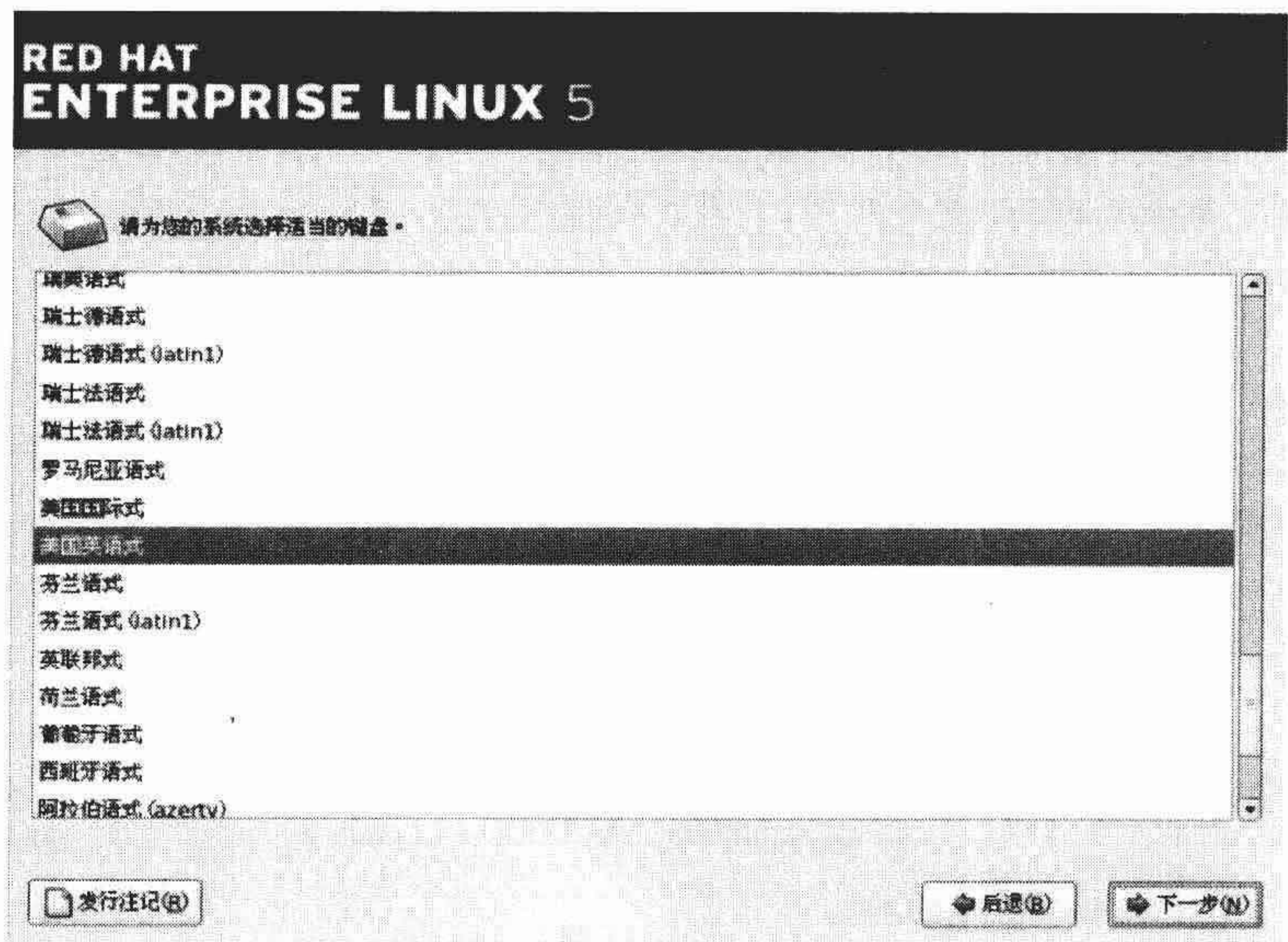


图 2-4 选择所需的键盘配置

提示：如果你使用的是 Gnome 桌面（有关桌面的更多信息见第 3 章），可以打开“应用程序”菜单（在面板上的主菜单里），单击“系统配置”标签，然后选择“键盘”应用程序修改键盘选项。在命令行里可以运行 `system-config-keyboard` 命令。

下一画面需要输入 Red Hat 安装号码。安装号码是在购买 Red Hat 时提供的。如果订阅

了 Red Hat 但找不到安装号码，可以到 Red Hat 支持站点找到它：<https://www.redhat.com/wapps/support/protected/subscriptions.html>。图 2-5 显示了安装号码输入屏幕。

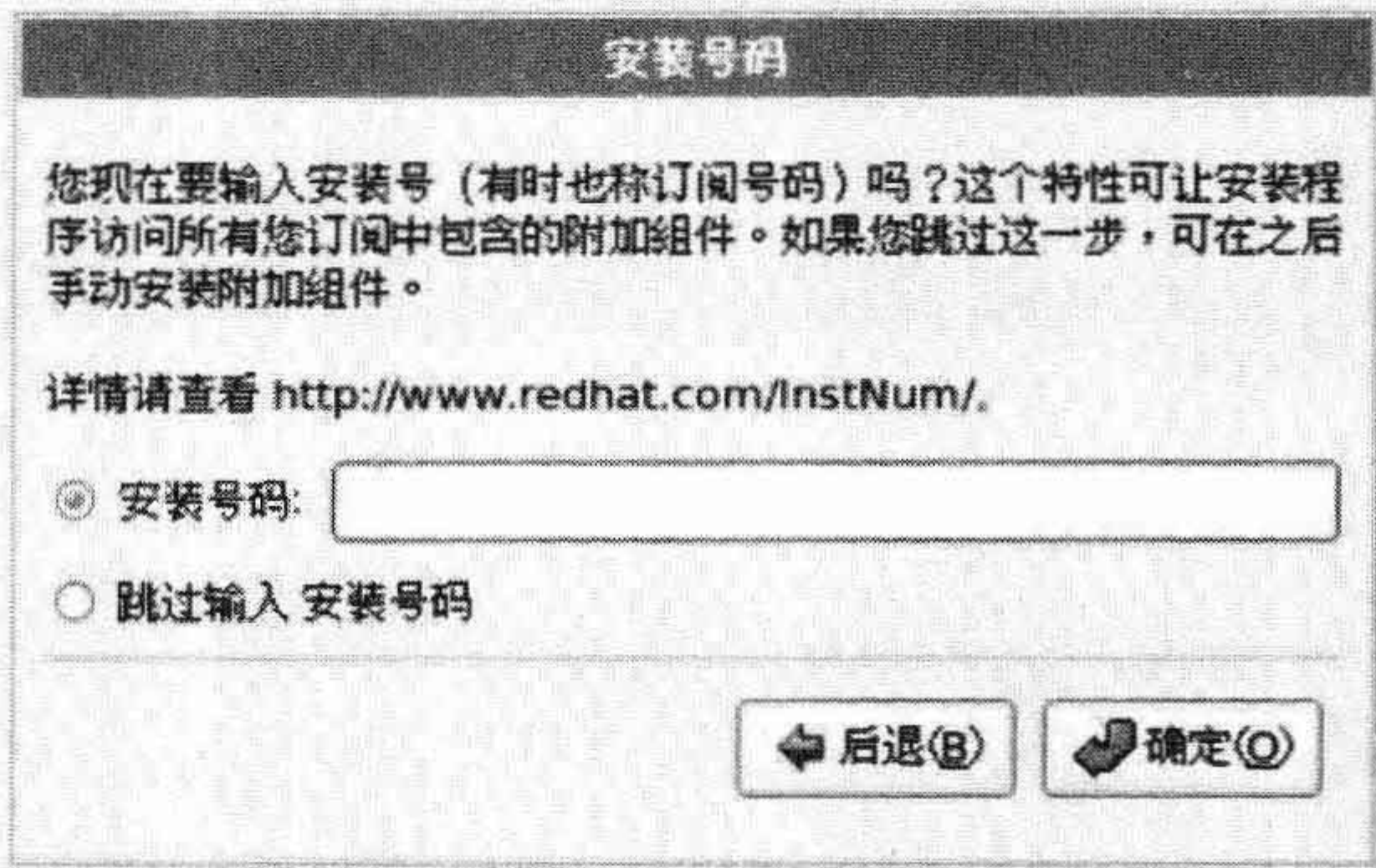


图 2-5 在此处输入安装号码

注： 如果不想为 Linux 花钱，那么像 CentOS、Fedora 或者 Ubuntu 这样的发行版也许是更好的选择，这强于没有补丁程序和更新的 Red Hat Enterprise Linux。

假定你购买了它并已经输入安装号码。如果没有购买，仍然可以继续，以后再提供安装号码，或者使用 Red Hat Enterprise Linux 服务器，但在购买它之前不进行任何更新。推荐购买，这样就可以得到最新的补丁程序和安全修补更新。

注意： 接下来的几步可能有危险。如果主机上已经存在操作系统或者重要数据，就可能失去所有数据，操作系统可能变得不可用。往下进行时请保持适度的谨慎，如果需要，还要有必要的备份制度。

在安装号码之后，接下来很可能会收到一个警告，告诉你即将创建一个新的分区表。这在全新的主机上是正常的，而且是我们所希望的，因此选择“是”继续。如果主机已经存在一个以前安装的操作系统，或者要与别的操作系统，如 Microsoft Windows，共享此主机，那么下面一步就不是希望有的步骤，这是退出此过程的时机。如果你认为会毁坏上次安装的有用数据，请选择“否”退出安装。

注： 如果正在覆盖已有的 RHEL 安装或者安装虚拟机，将不会看到上述警告。

图 2-6 显示了分区的初始化过程。

新分区表创建完成后，接下来为系统创建分区。为磁盘分区就像切蛋糕，可根据消费者的胃口选择磁盘每一分区的大小。例如，如果系统里有一个网站并且该网站有相当多的日志，那么就要注意磁盘的划分，要让保存这些网站数据和日志文件的分区有更多的空间。如果要运行一个文件服务器，那么就要保留更多的磁盘空间给用户数据而不是给网站数据或日志。

注： 第 8 章将阐述更多有关分区以及如何定制和修改磁盘与存储的内容。



图 2-6 通过创建新的分区表对驱动器初始化

你可以看到它列出了可用的硬盘列表，现在可以挑选要用的磁盘。我们的情况是只有一个可用磁盘，因此就选择默认的“在选定驱动上删除 linux 分区并创建默认的分

区结构”选项。与此同时，选中“检验和修改分区方案”选项，这样就可看到这些默认设置组合的效果了。如图 2-7 所示。

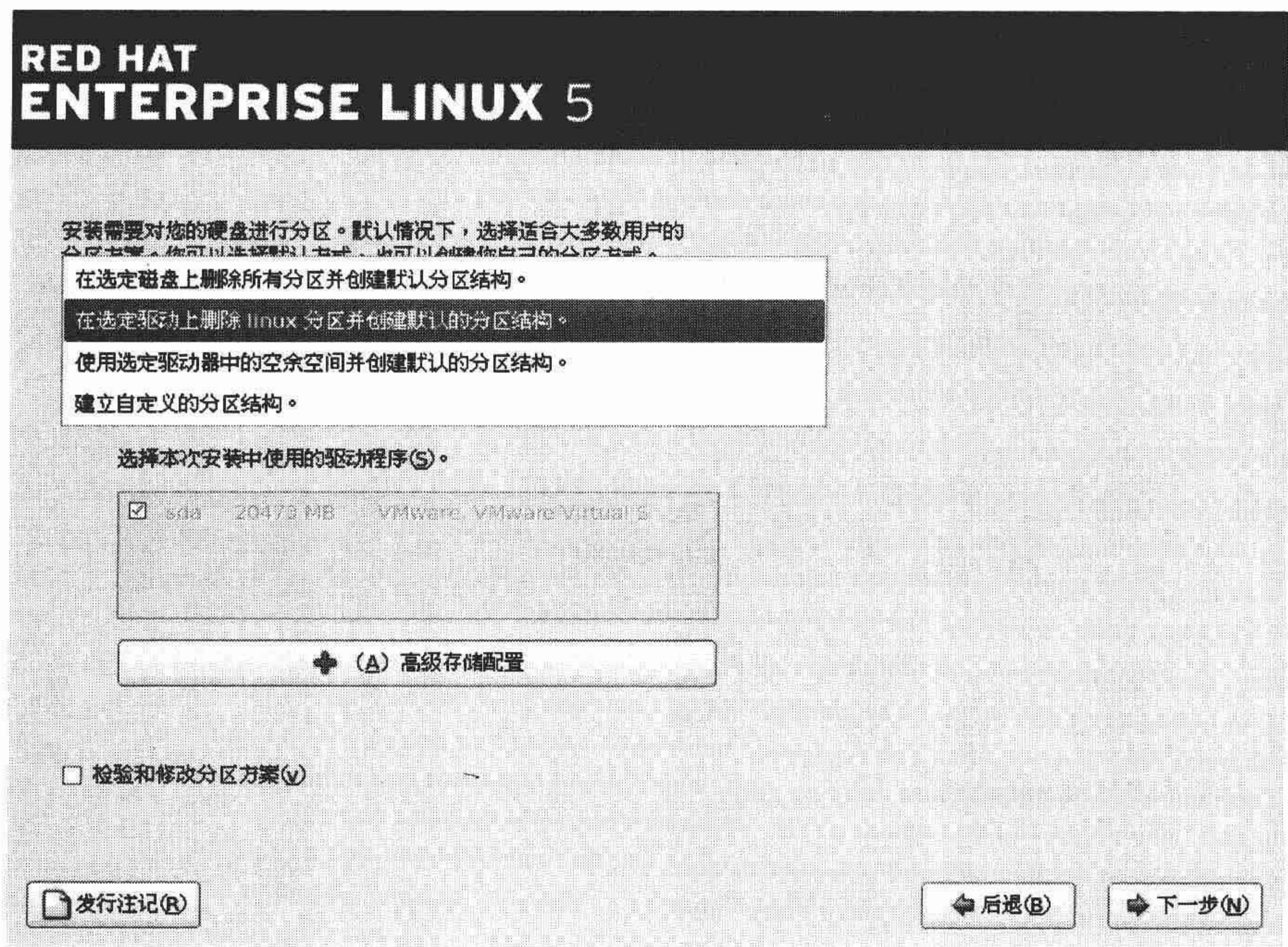


图 2-7 选择分区布局

现在会看到另一个警告，这次它告诉你正在从驱动器移除所有的 Linux 分区，是否要继

续，如图 2-8 所示。选择“是”。

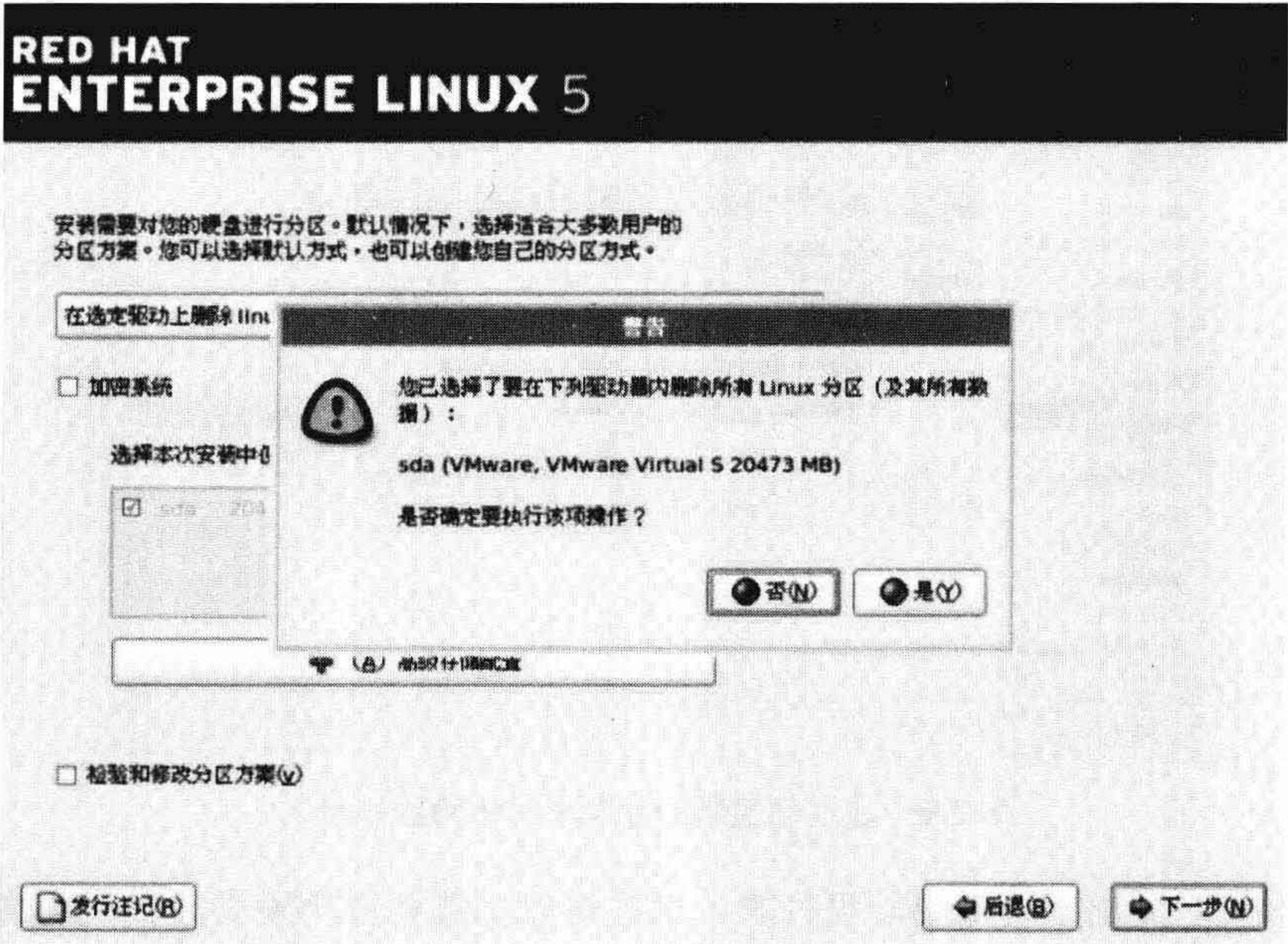


图 2-8 移除分区的警告

最后，图 2-9 显示了分区方案的布局。

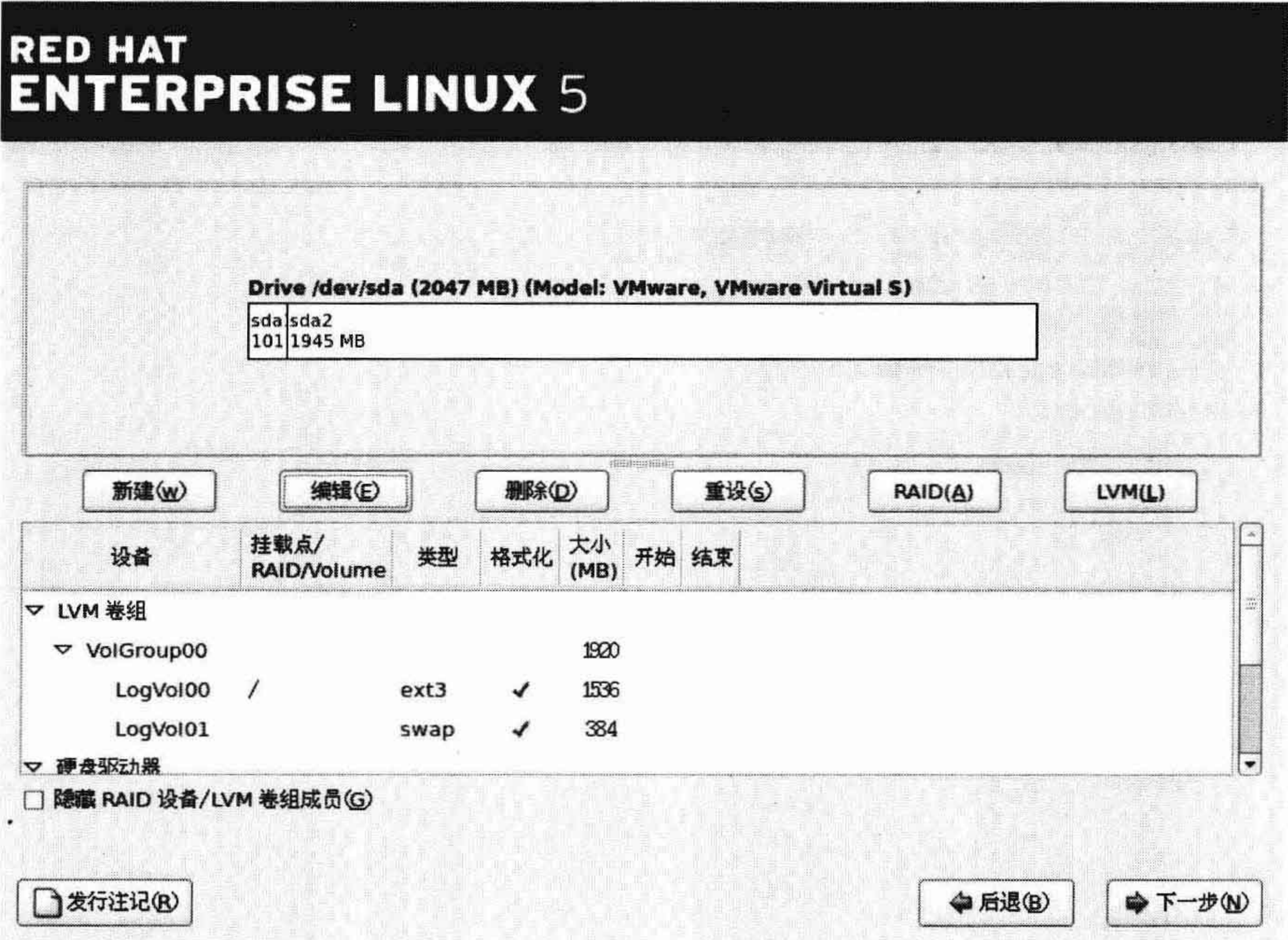


图 2-9 复查分区布局

正如你所看到的那样，默认做法是使用逻辑卷管理（logical volume management, LVM）

工具创建一个拥有两个逻辑卷的组。

■注：使用 LVM 可以管理磁盘卷以及在线扩展、缩减和修改磁盘分区的大小。LVM 软件使管理员不必重建系统和重新格式化就可修改磁盘布局，添加新磁盘存储或者把磁盘存储移除并重新用作系统的其他部分。第 8 章将详述 LVM。

默认情况下，第一个卷被称作 swap。swap 卷是用来保存超出 RAM 容量那部分数据的磁盘空间。它一般为 RAM 容量的两倍。例如，如果 RAM 容量为 512MB，那么 swap 卷的大小为 1GB。第二个卷被称作 root 或卷。该分区保存所有主机数据和应用程序。尽管图 2-9 没有清楚地显示出来，其实还有一个 100MB 的分区叫做/boot。/boot 分区保存 Linux 内核以及主机启动所需的一些代码。

如果有任何特别的需求，可以改变这种默认结构，创建新的分区，或者删除所有的东西并使用安装程序的分区管理器重新开始。我们将在第 8 章深入介绍如何划分一个或几个各种用途的驱动器。

下一画面是 GRUB (Grand Unified Bootloader) 引导程序的安装画面。引导程序是计算机开启后紧接着 BIOS 运行的那段程序，它默认安装在主硬盘驱动器的头 512 字节（主引导记录或者 MBR）。GRUB 用来在打开计算机时引导 Linux 内核。如果重启主机，首先碰到的就是 GRUB，并且在启动时可以看到它提供的菜单。

■注：在某些情况下，你可能希望在别的分区安装引导程序。例如，你可能运行一个有“双启动”功能的主机，意思就是主机可引导两个或更多不同的操作系统，像 Windows 和 RHEL，但不是同时引导。在这种情况下，MBR 会被别的操作系统重写，那将暂时失去引导 Linux 的功能。在本章稍后的图 2-20 中，可以看到主机启动时出现的 GRUB 菜单。在 http://apcmag.com/the_definitive_dualbooting_guide_linux_vista_and_xp_stepbystep.htm 上可以找到一个讲述双启动的指南。

在图 2-10 中的画面中选择希望引导程序安装的位置。我们选择了唯一可用的磁盘位置/dev/sda。

■注：可以使用多种磁盘类型作为存储硬件。不同的磁盘有不同的名字。图 2-10 显示了驱动器/dev/sda，其中 s 代表 SCSI 或者 SATA。IDE 磁盘以 h 为前缀（例如/dev/hda）。不过，最新的内核版本使用较新的 IDE 驱动器，为了一致，它以/dev/sd*为 IDE 磁盘命名。

你还可以为引导程序设置一个密码，以保护启动选项不被修改。比方说，如果有人想对 GRUB 启动菜单做改动，则必须提供正确的密码。对于那些同在一个数据中心而易于访问的主机或者位于其他物理安全性差的地方的主机来说，这通常是个不错的主意。

■注：物理安全对于 Linux 主机来说是重要的——你不希望任何人窃取有价值的物理资产（和数据）。你应该把主机放到带锁的箱子或机柜，或者可控制访问的房间里。如果把服务器放在一个共管的地方或者数据中心，那么要确保这个地方有妥善的物理安全控制机构以保护主机。



图 2-10 在/dev/sda 上安装引导程序

引导程序高级选项允许指定引导程序的安装位置，修改驱动器的次序以及设置一些类似 Force LBA32 的历史遗留选项。如果需要在启动期间传递任何内核参数，也可以在这里添加它们。一般不必修改这些选项，所以我们只选择默认值。

如图 2-10 所示，启动进入的默认操作系统已经选定，它是 Red Hat Enterprise Linux 服务器。引导程序安装之后，现在要配置网络。下一画面（见图 2-11）显示的是网络配置。

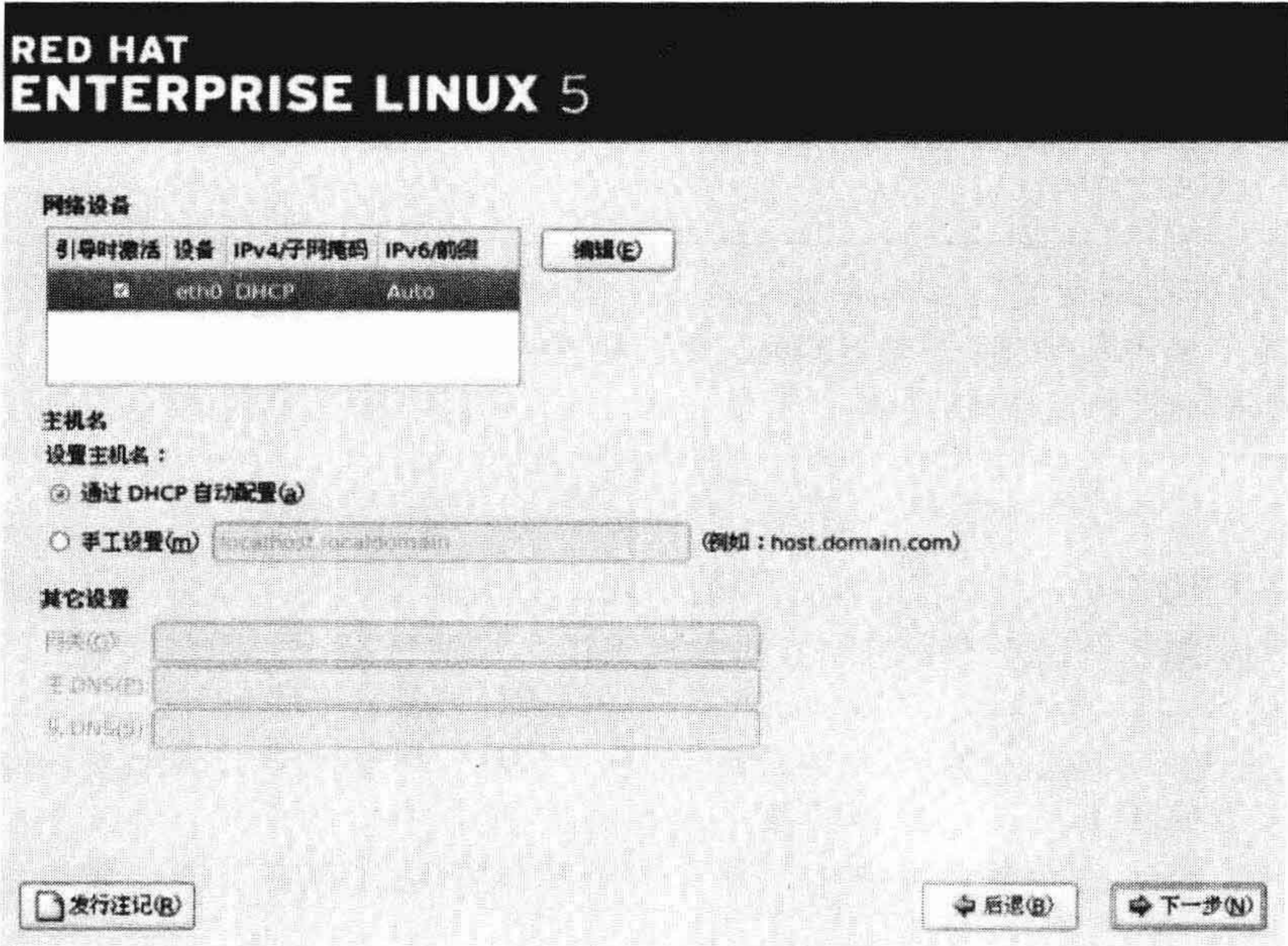


图 2-11 网络设置

这里最重要的配置项是 IP 地址，它是主机的网络地址，其他主机通过该地址可以找到它并与之通信。

■提示：关于 IP 地址和编址的信息，可访问 http://en.wikipedia.org/wiki/IP_address 和 <http://computer.howstuffworks.com/question549.htm>。

网络 IP 地址的分配方式一般有两种。第一种方式是在配置时直接指定每个主机的 IP 地址。这被称为静态地址。第二种方式使用一种被称为动态主机配置协议（Dynamic Host Configuration Protocol, DHCP）的网络服务。DHCP 利用网络上的服务器，在主机需要的时候为他们分配 IP 地址。DHCP 服务器监视这些地址并确保没有冲突。你可能已经配有一个具有 DHCP 功能的路由器（正如大多数 ADSL 调制解调器/路由器那样）。在下一画面，你可以选择 DHCP 选项，以便自动获得 IP 地址。

■注：第 9 章将介绍如何配置自己的 DHCP 服务器。

不过，考虑到这次安装的目的，我们将通过选择“编辑”来设置一个静态 IP 地址。图 2-12 指定了一个适合我们网络的地址。你应该根据你的环境输入适当的配置信息。完成后，选择“确定”。

■提示：图 2-12 显示了一个 IPv6 的复选框，IPv6 是另一种较新的 IP 地址格式，还没有得到广泛应用。关于 IPv6 的内容可到 <http://en.wikipedia.org/wiki/IPv6> 了解。大多数 Linux 发行版都支持这种新的编址格式。

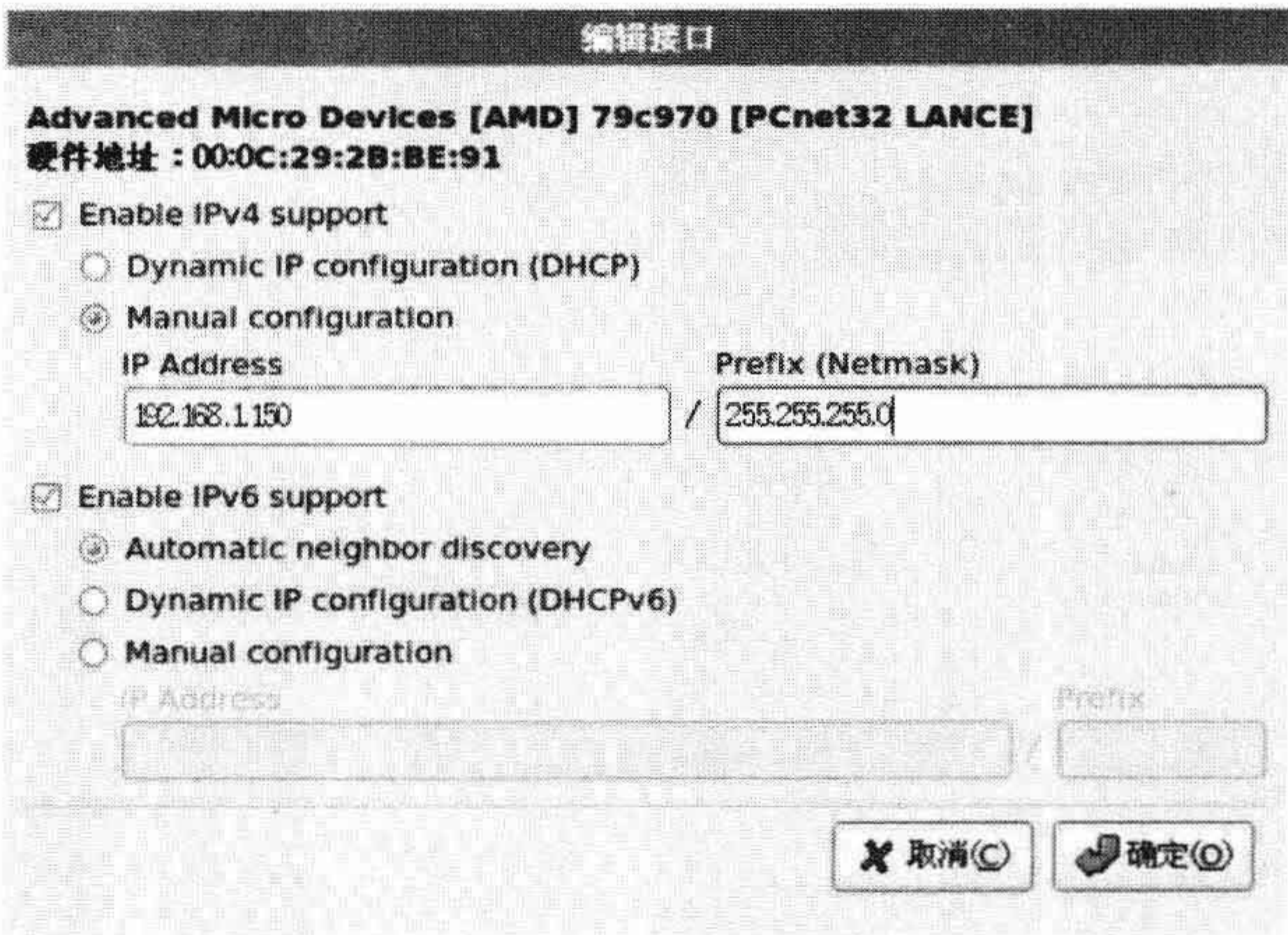


图 2-12 设置 IP 地址

设置好 IP 地址后，现在需要为主机添加一个名字，或叫主机名。在初创公司中有一种趋势，以特别喜欢的电视角色、乐队或者神话中的生物名作为主机名。虽然这很有趣，但当完成特定任务的多台主机位于多个地理位置时，很快就会有麻烦。我们的主机名是 au-mel-rhel-1，因为我们更喜欢描述性的命名标准：地区-城市-操作系统类型-序号。再举一个描述性命名格式的例子，如果有一台文件服务器在美国，IP 地址结尾为 155，就可以选择

us-ny-fileserver-155 这个名字。关键问题是描述性，不要把主机命名为“Brittany”或者“Thor”。

■注：可以选择任何你喜欢的适合你环境的命名标准。我们更偏好描述性的命名规则。

另外还需指定一个默认网关和一个或多个域名系统（Domain Name System，DNS）服务器。默认网关是所有的流量离开我们的网络之前必经的路由点。它或者是一个调制解调器/ADSL 网关，或者是一个把你的网络连接到 Internet 或者其他私有网络的物理路由器。第一和第二 DNS 服务器是把 IP 地址解析为完全合格域名的专用服务器。

■提示：指定第一和第二 DNS 服务器增加了网络的冗余度。如果一个服务器没有响应，主机尝试另一个服务器。

主机每次访问网站时利用默认网关和 DNS 服务器找出到达那儿的途径。例如，如果在浏览器的地址栏键入 `www.google.com`，主机会首先找 DNS 服务器，它可能在或者不在你的网络上。如果不在你的网络上，主机会利用默认网关到达那里。接着主机向 DNS 服务器查询 `www.google.com` 的 IP 地址，DNS 服务器会用类似 `74.125.19.104` 的信息作为答复。然后主机再利用默认网关离开我们的网络，然后抓取由 `www.google.com` 提供的网页。通俗地说，DNS 服务器就是地图，默认网关是寻找你想要的东西要走的第一条街。

图 2-13 所示选择了适合我们网络的设置。这些服务和设置将在第 9 章讲述如何设置 DNS 服务器和管理路由器时更详细地讨论。

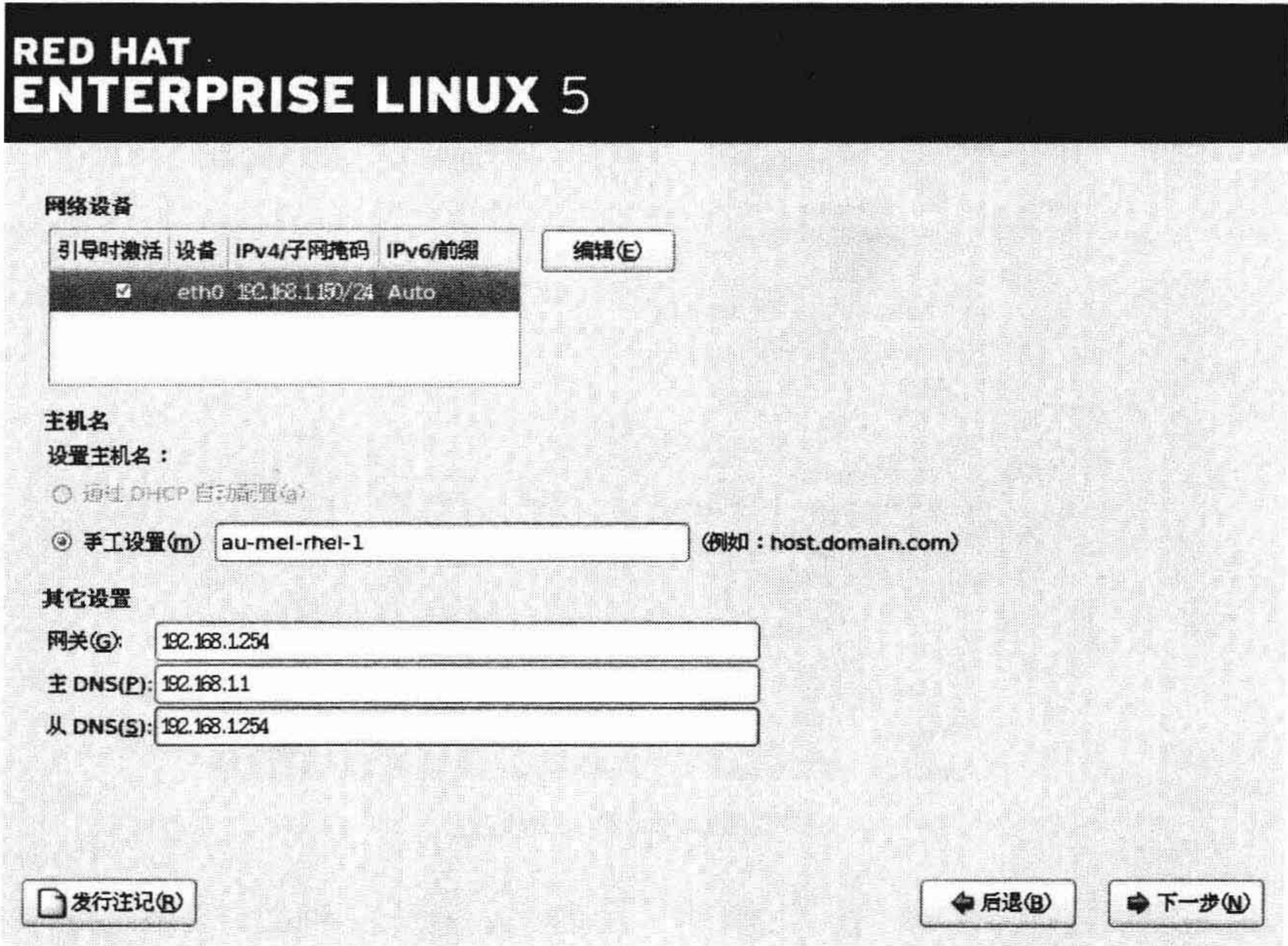


图 2-13 主机名、默认网关和 DNS 设置

下一个画面是一个不错的测试，看看你是否知道自己相对于这个星球上的其他人所处的位置。使用鼠标指到离你最近的主要城市的黄点上以便设置正确的时区，或者从下拉列表中

选择区域。图 2-14 选择了上海（原文为“澳大利亚的墨尔本”）。

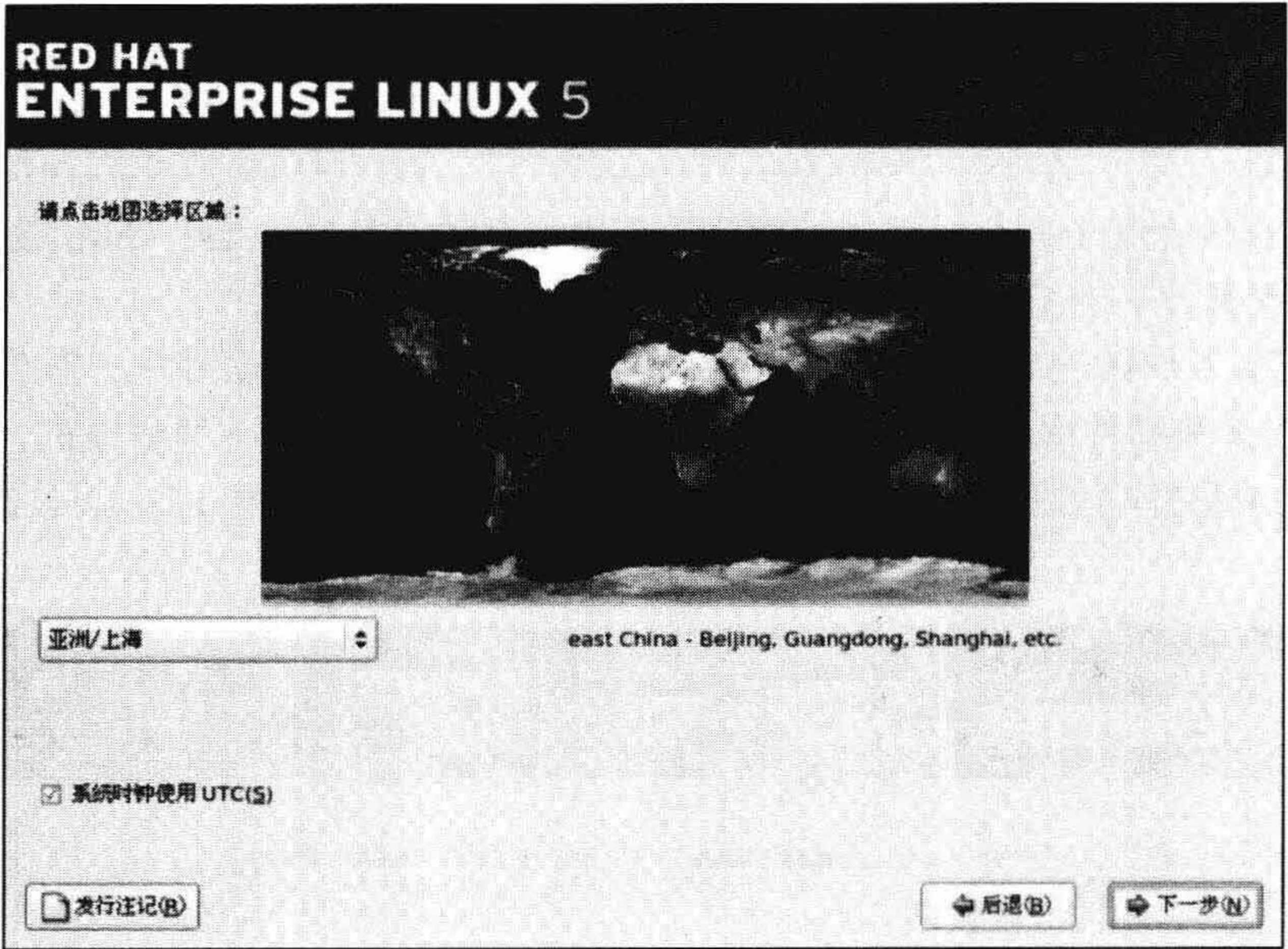


图 2-14 选择位置

接下来要求输入 root 密码，如图 2-15 所示。在 Linux 上，root 用户是有权限访问整个系统的超级用户，很像 Windows 操作系统中的 Windows 管理员。

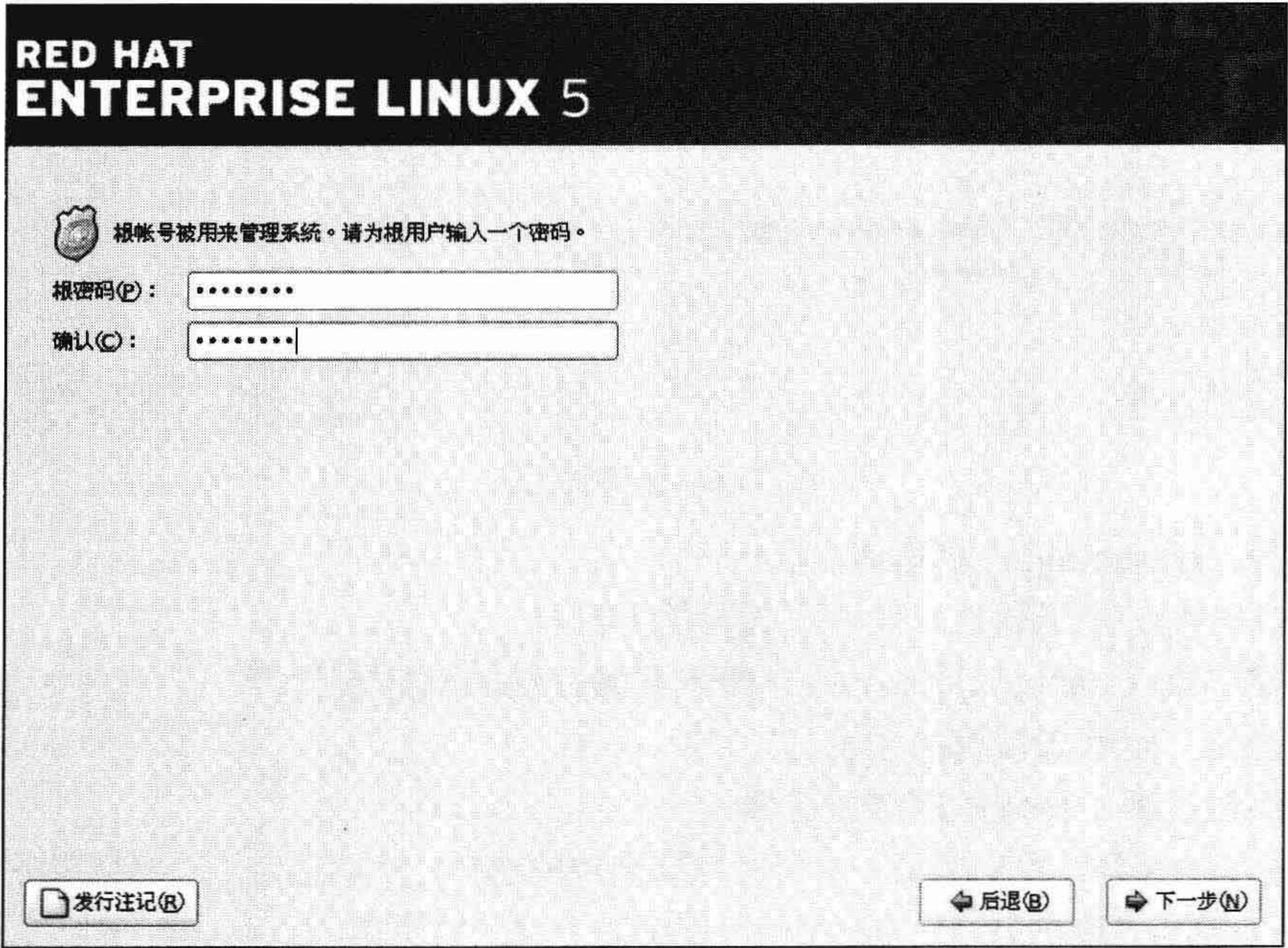


图 2-15 root 密码

■提示：第 3 章和第 4 章将详细讨论 root 用户。

这个密码应该是复杂的，应该由大小写字母、数字和@!%#*之类的特殊符号等组合而成。另外，它的长度至少应该为 8 个字符。

■提示：可以到 http://en.wikipedia.org/wiki/Password_strength 了解一个好密码的特征。

把该密码存放在某个安全的地方。你可以使用如开源产品 KeePass (<http://keepass.info/>) 这样的密码安全程序，把它存放在加密的 USB 便携存储设备中。

所有的准备工作都已完成，现在就可以考虑选择想要安装的软件包。我们将安装能够运行网络、DNS 和邮件服务器的软件包。RHEL 的安装过程是非常细粒度的，可以通过指定主机的角色来选择功能和应用程序，非常接近个性化的程序。图 2-16 显示了主机的一些可能角色。

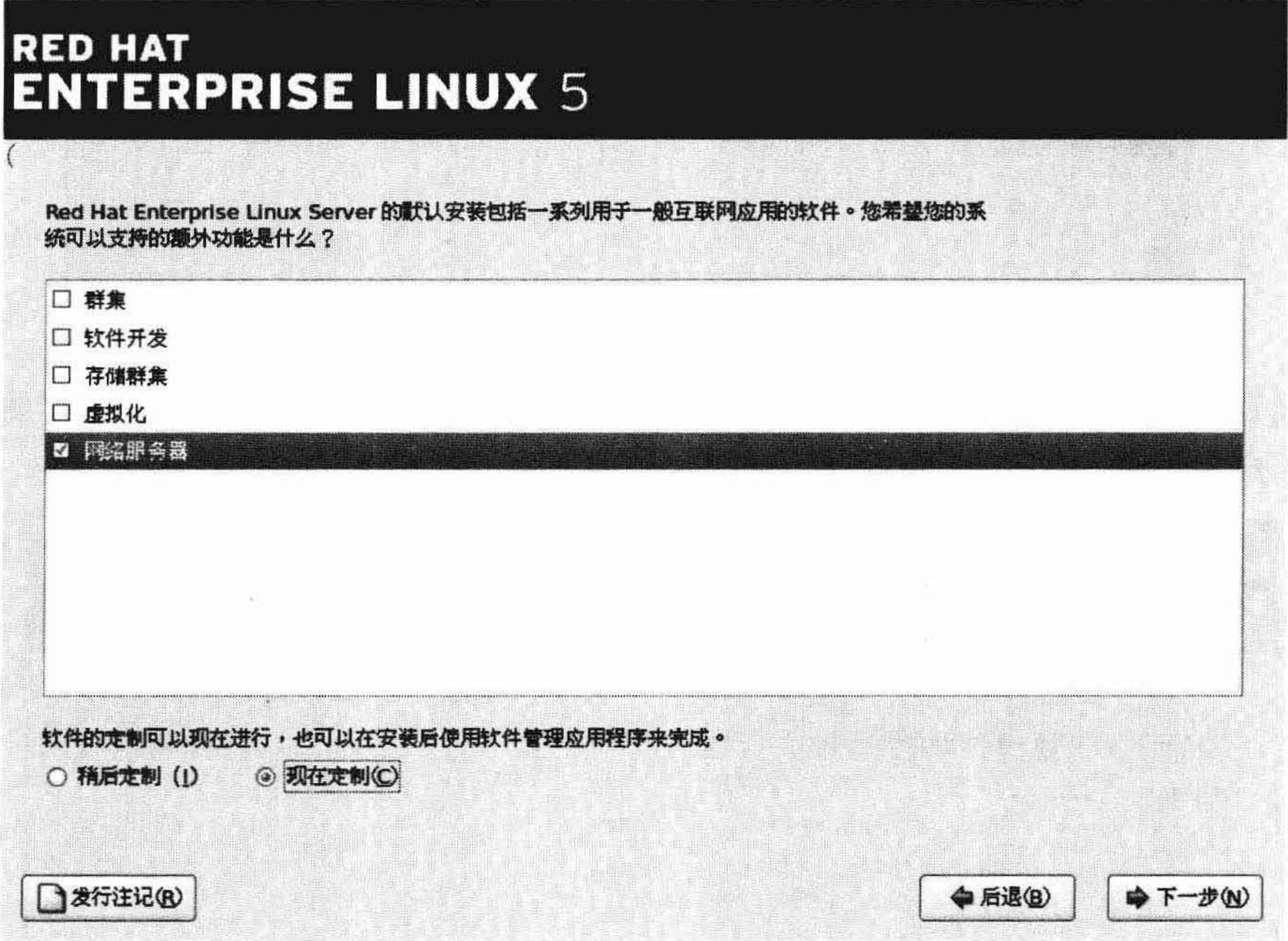


图 2-16 选择软件包

- 群集 (clustering)：机群成员。
- 软件开发：用于软件开发和写代码的主机。
- 存储群集 (storage clustering)：提供群集存储的机群成员。
- 虚拟化：运行虚拟机的主机。
- 网络服务器：传统的网络服务器。

正如所看到的那样，我们取消了对所有默认选项的选择，然后选定了“网络服务器”和“现在定制”选项。选定“现在定制”单选按钮可以练习更深入的操作，以及选择个性化的应用程序。

下一画面显示了可供选择安装的所有不同软件包组。当看到“可选的软件包”按钮时，你可以单击它微调这些软件包。使用左边框架选定感兴趣的组类型（应用程序、开发等），然后使用右边框架选定组。图 2-17 打开了“服务器”标签，并选择安装 DNS 名称服务器、邮

件服务器和 MySQL 数据库。

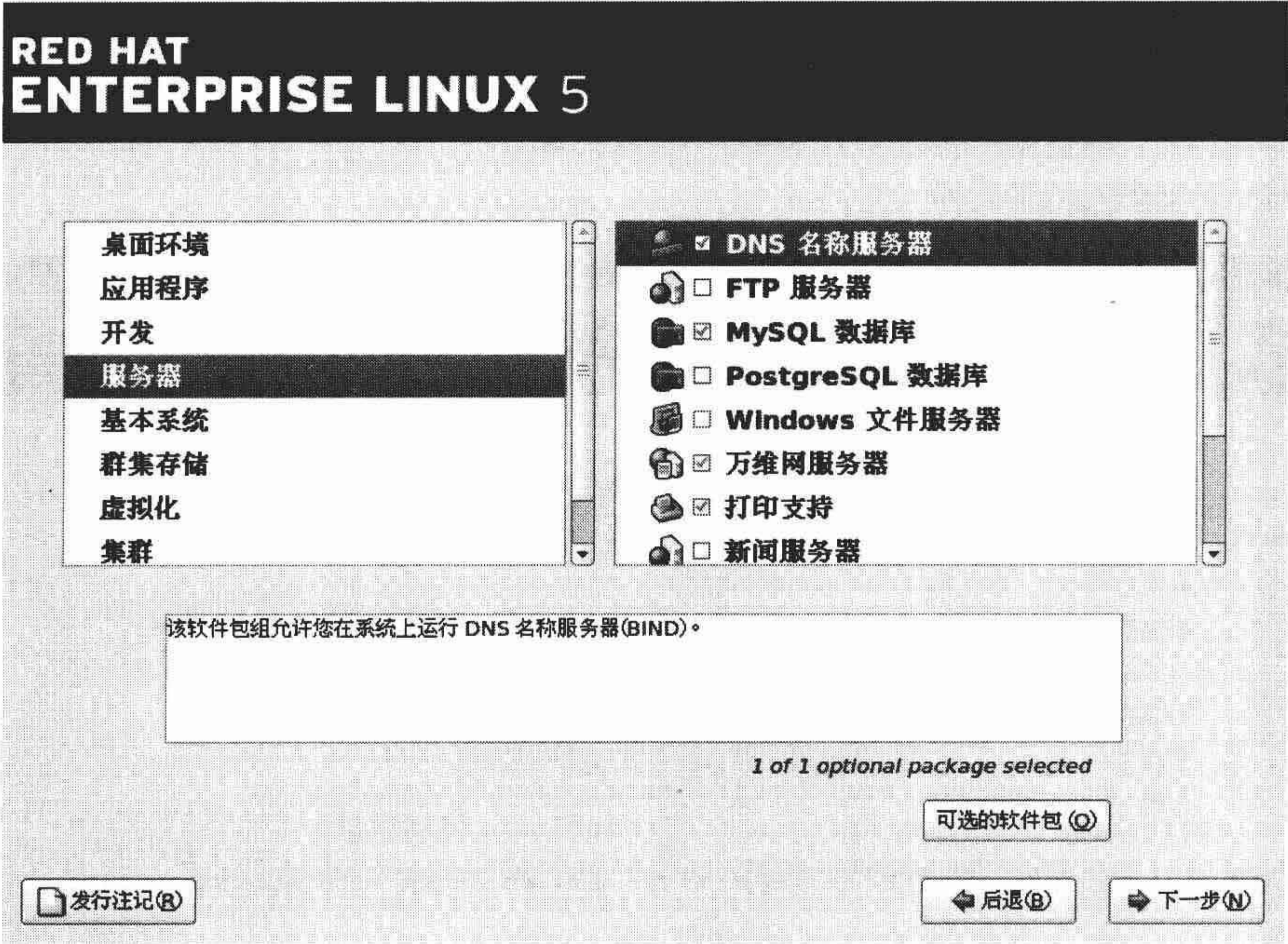


图 2-17 从 Servers 列表中选择软件包

如果你单击 MySQL 数据库下的“可选的软件包”按钮，可以看到另外 11 个软件包（图 2-18 显示了其中一部分），可以根据个人喜好选择不选择它们。

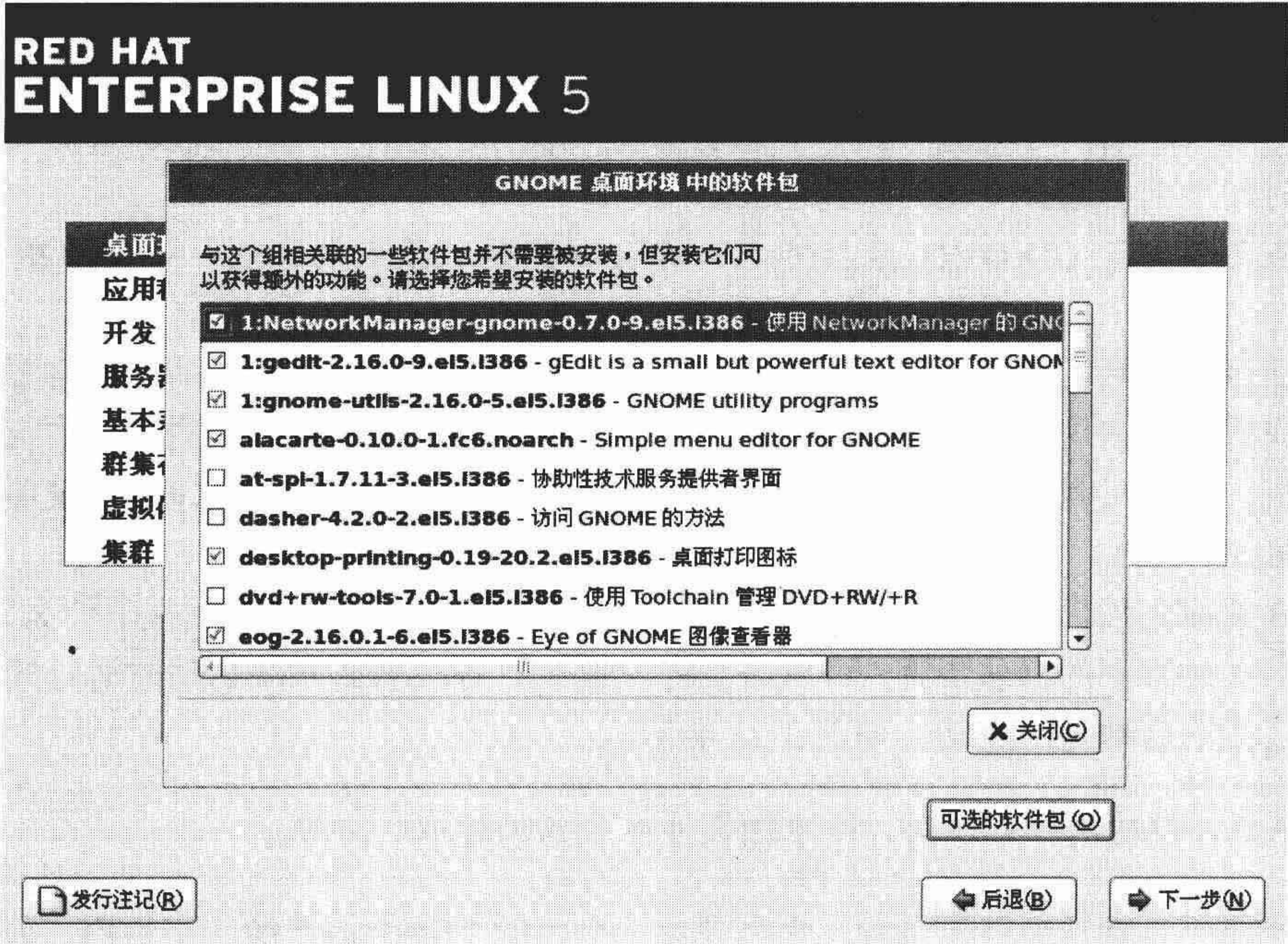


图 2-18 选择可选软件包

你准备得差不多就可以开始往磁盘写入安装程序了。在这个过程的第一步里，安装程序将检查包依赖问题。它扫描已选择的软件包，并确定为支持已选软件包而需要安装的附加软件包。

检查完成后，安装程序会确保有足够的磁盘空间来安装所选软件包。最后会呈现安装准备就绪的画面，如图 2-19 所示。

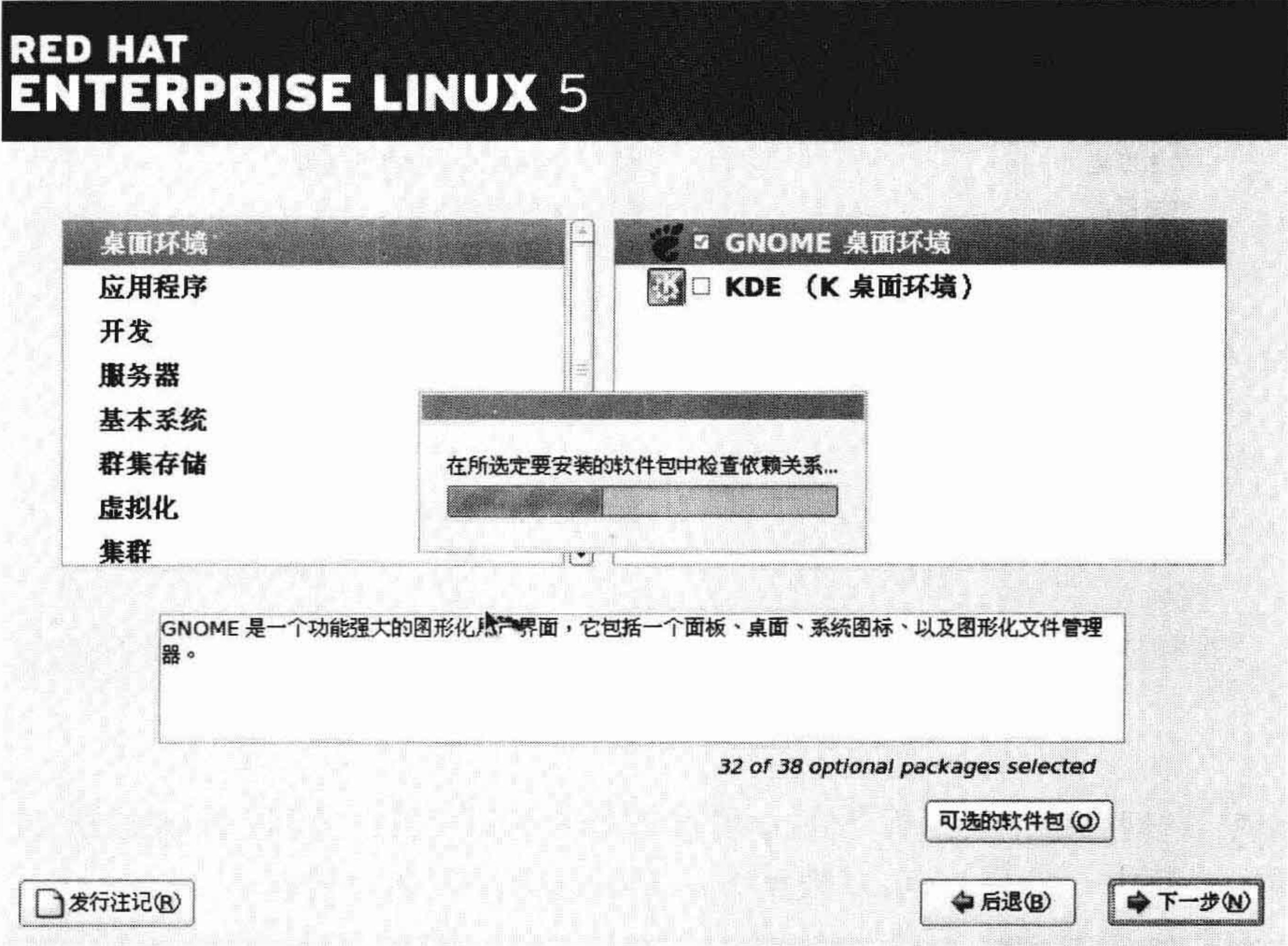


图 2-19 安装准备就绪

安装程序一启动，你就能看到有一个显示安装状态的进度条。安装过程本身需要花费数分钟，具体时间根据所选择安装的软件包数量而定。

一旦安装完成，CD/DVD 就会弹出，你会看到要求重启的画面。单击“重启”重启主机。

重启后可以看到 GRUB 菜单画面（记得吧，这是早先安装的引导程序），它要求选择进入哪个操作系统。在这一画面什么都不必做，因为几秒后它会选择默认的操作系统并开始启动新的主机。如果在倒计时期间按了任意键，加载就会被打断，完整的 GRUB 菜单会显示在屏幕上（见图 2-20）。

GRUB 完成之后，呈现的是 Red Hat 的闪屏。你可能想知道“它现在在干什么？”。为了查看主机启动过程中的具体步骤，可单击“显示细节”。你会看到系统启动时运行的所有进程以及这些进程的启动是否成功，如图 2-21 所示。

注：关于启动过程的详细讲解，请参阅第 5 章“启动过程”一节。

如果选择的是默认软件包选项，它会安装一个叫做“firstboot”的应用程序。因为这是第一次启动主机，所以 firstboot 程序会启动，它会提出一系列有关主机以及如何配置它的问题。

在此期间，还有机会注册 Red Hat 网络（Red Hat Network，RHN）。

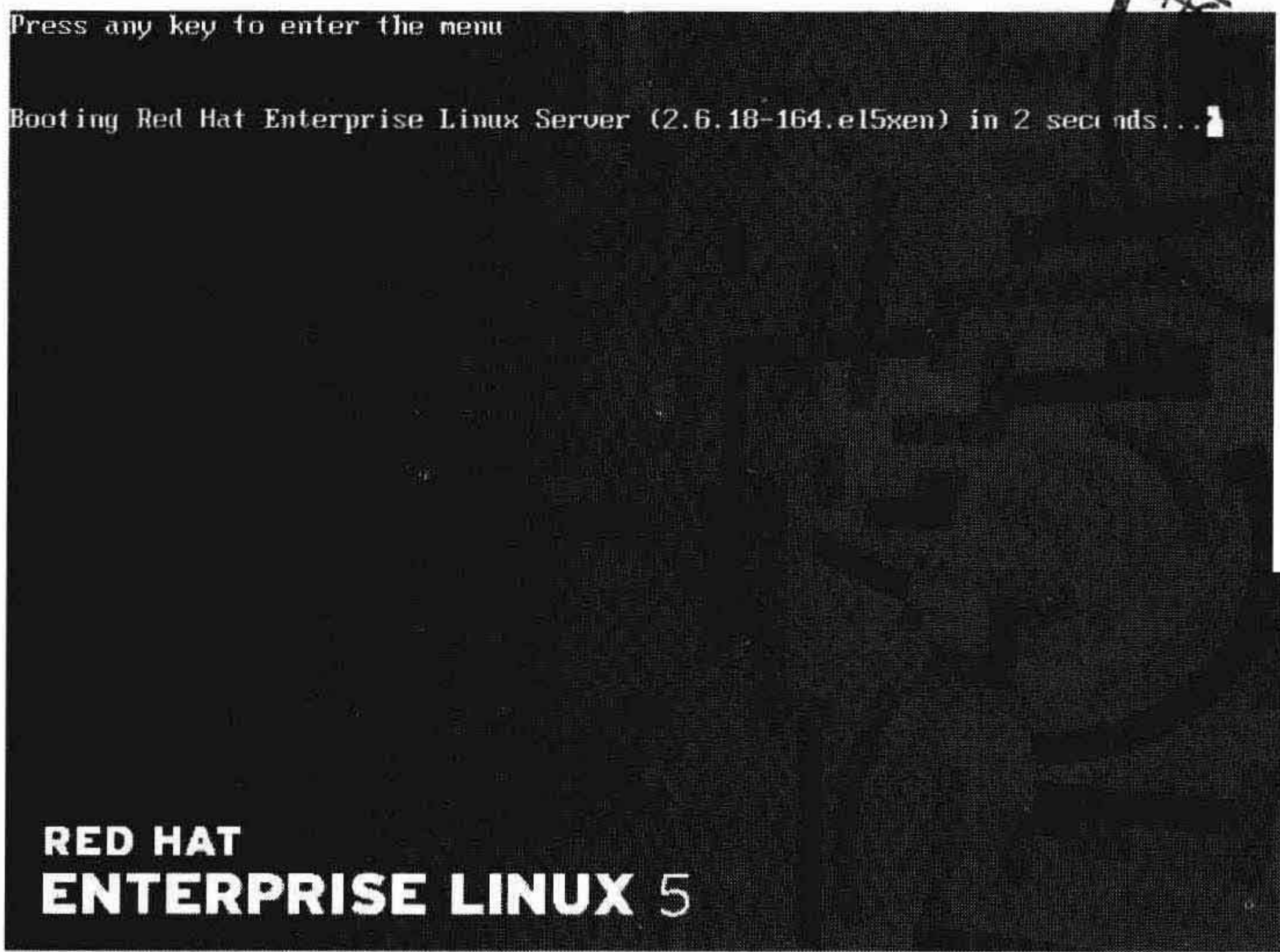


图 2-20 启动新主机

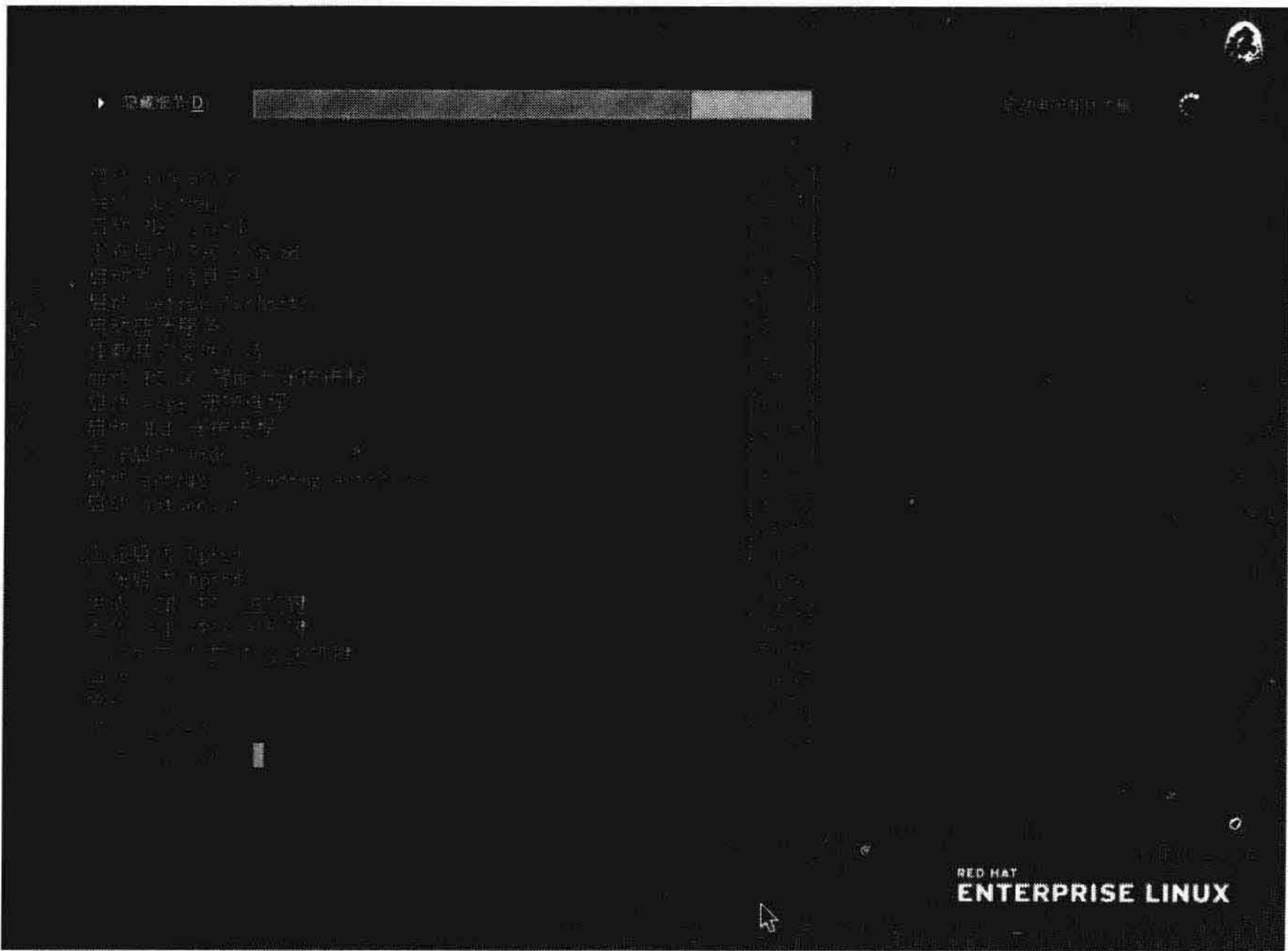


图 2-21 显示系统启动状态

首先看到的是图 2-22 所示的“欢迎”画面。单击“前进”继续。

接下来它要求阅读并同意许可协议。如果同意遵守协议，选择“是的，我同意”；如果不同意，选择“不，我不同意”。如果不同意，你可以选择重读许可协议，或者结束安装；只有同意之后才可以继续使用主机。

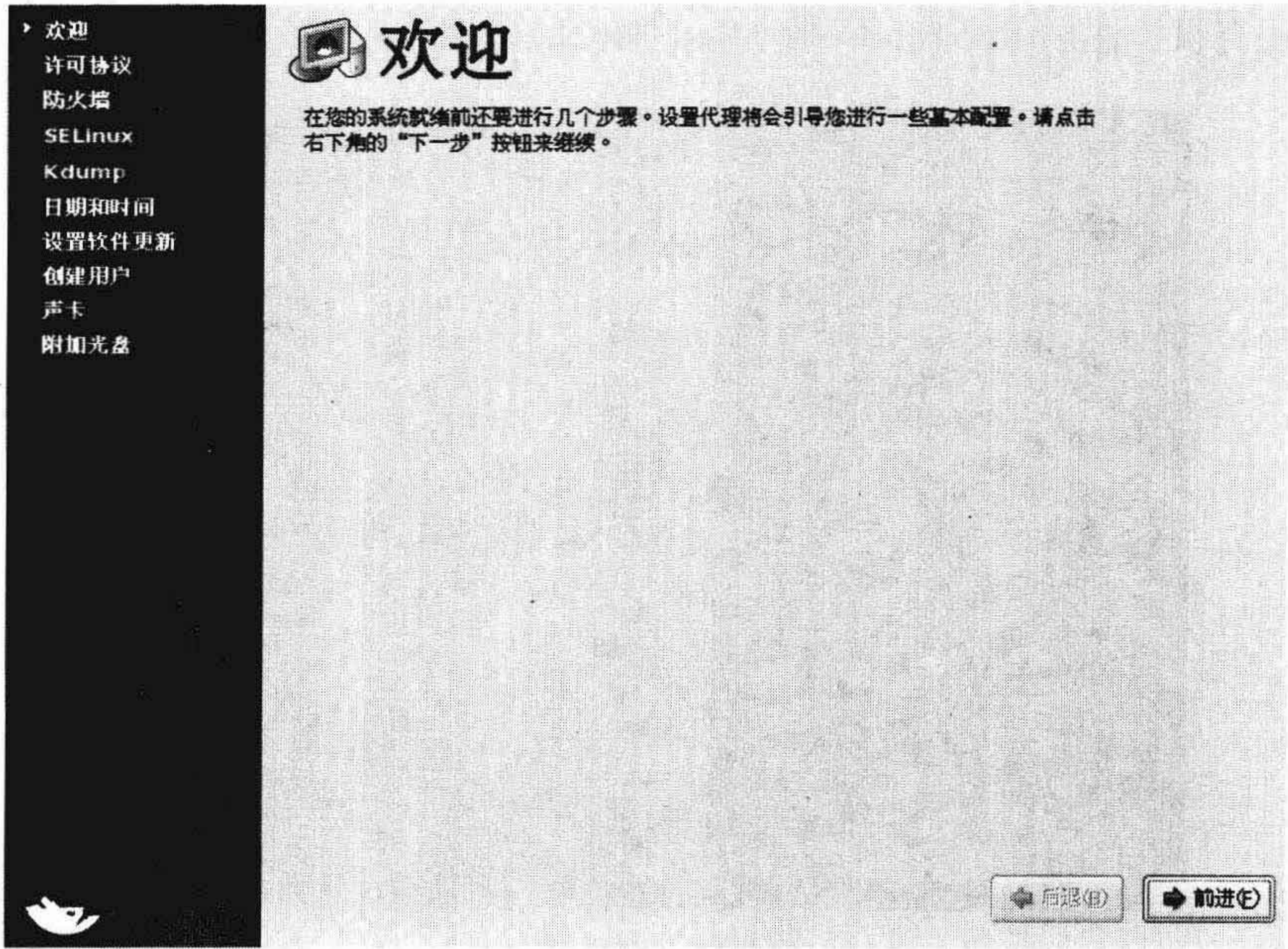


图 2-22 “欢迎”画面

许可协议画面之后，它会提示配置主机防火墙。大多数 Linux 发行版配置和安装一个基于主机的防火墙，它使用 Netfilter 防火墙（也就是大家所知的 iptables 防火墙，它以配置该防火墙的命令而得名）。我们推荐在所有主机上打开这个防火墙。

现在仅仅配置防火墙允许 TCP 包在端口 22 传输，这样就可使用一个称为安全外壳（Secure Shell，SSH）的协议远程连接到主机。第 3 章和第 6 章中将详细阐述 SSH，在第 6 章还会讲到更多有关防火墙的知识。图 2-23 所示为我们的防火墙配置。

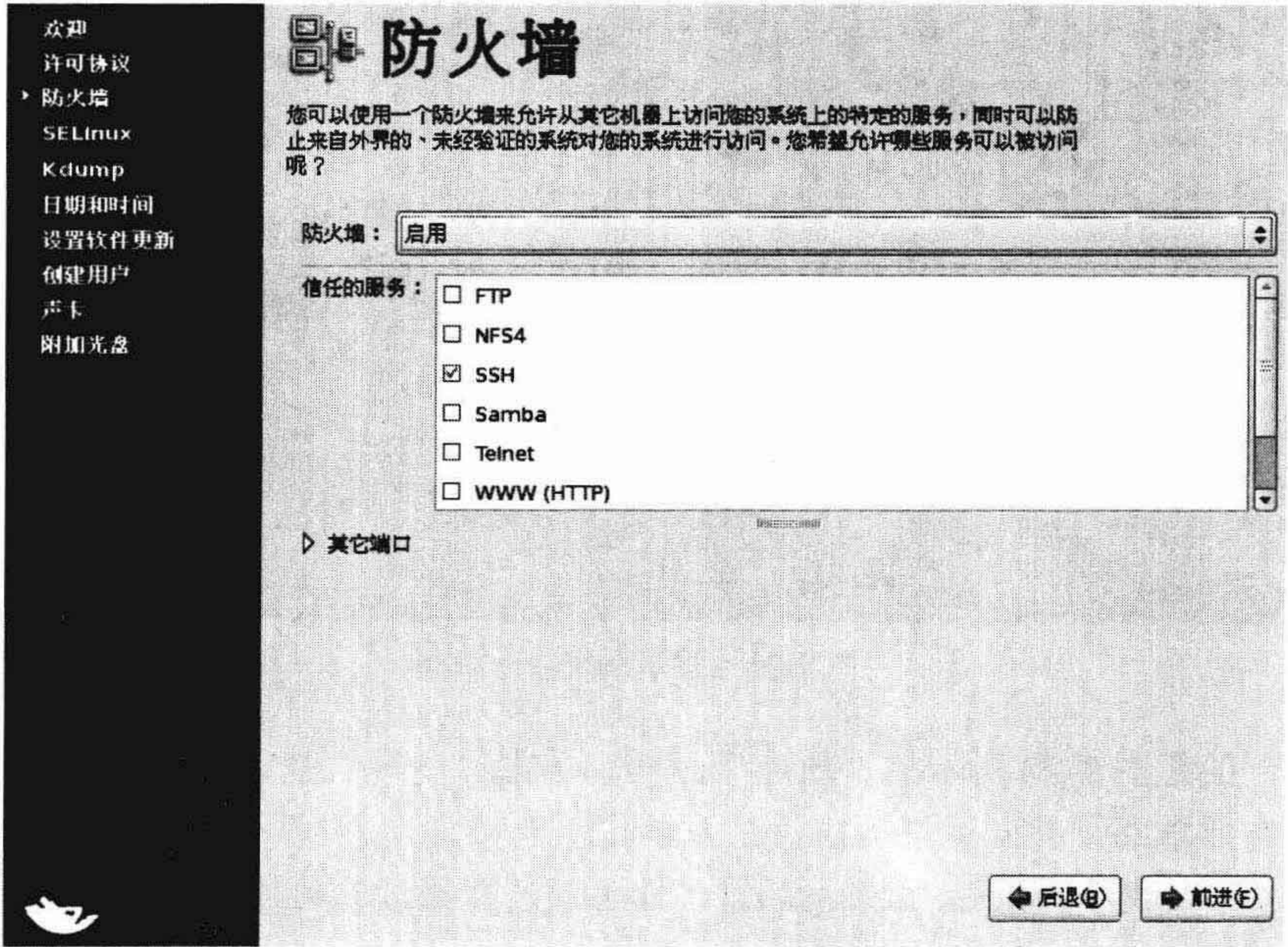


图 2-23 配置防火墙

下面的画面将提示为主机配置 SELinux 模式。SELinux 把名为强制访问控制（Mandatory Access Control, MAC）的安全控制应用到像文件、进程和信息这样的对象上。SELinux 是由美国国家安全局开发的，目的是“基于保密和完整性需求加强信息分离从而提供系统安全性”（<http://www.nsa.gov/selinux/index.shtml>）。

SELinux 是做什么用的？传统 Linux 遵循一个叫作自由访问控制（Discretionary Access Control, DAC）的概念。在 DAC 的概念下，一般的用户可以创建对象并给予权限（在第 3 和第 4 章你会了解到更多相关信息）。权限使得特定的用户或组能够读、写或者执行对象。如文件和应用程序之类的对象可以基于这些权限而相互作用。

在 SELinux 的 MAC 概念下，一组安全策略运行在系统级，这些策略控制系统上的所有对象，以及这些对象之间如何或者是否相互作用。那么举例来说，SELinux 下的 MAC 会阻止 Apache 网络服务器访问属于 Postfix 邮件服务器的文件，即使 Web 服务器受到损害而获得 root 访问权限也不例外。根据安全策略，这两个进程逻辑上独立，所以即使一个受损害了，另一个也不会连带受到损害。

因为刚开始使用新主机，所以设置模式为“允许”（见图 2-24），这意味着在没有实际实施的情况下，安全策略会把可能产生的问题都告知你。这给你提供了充分调整主机的机会，以便做好配置它的准备。在新主机试用之后，变成成品之前，我们应该把安全策略设置为“强制”，以使系统更安全。

■注：你可以到 http://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/Deployment_Guide-en-US/ch-selinux.html 了解如何在 Red Hat 上实现 SELinux。

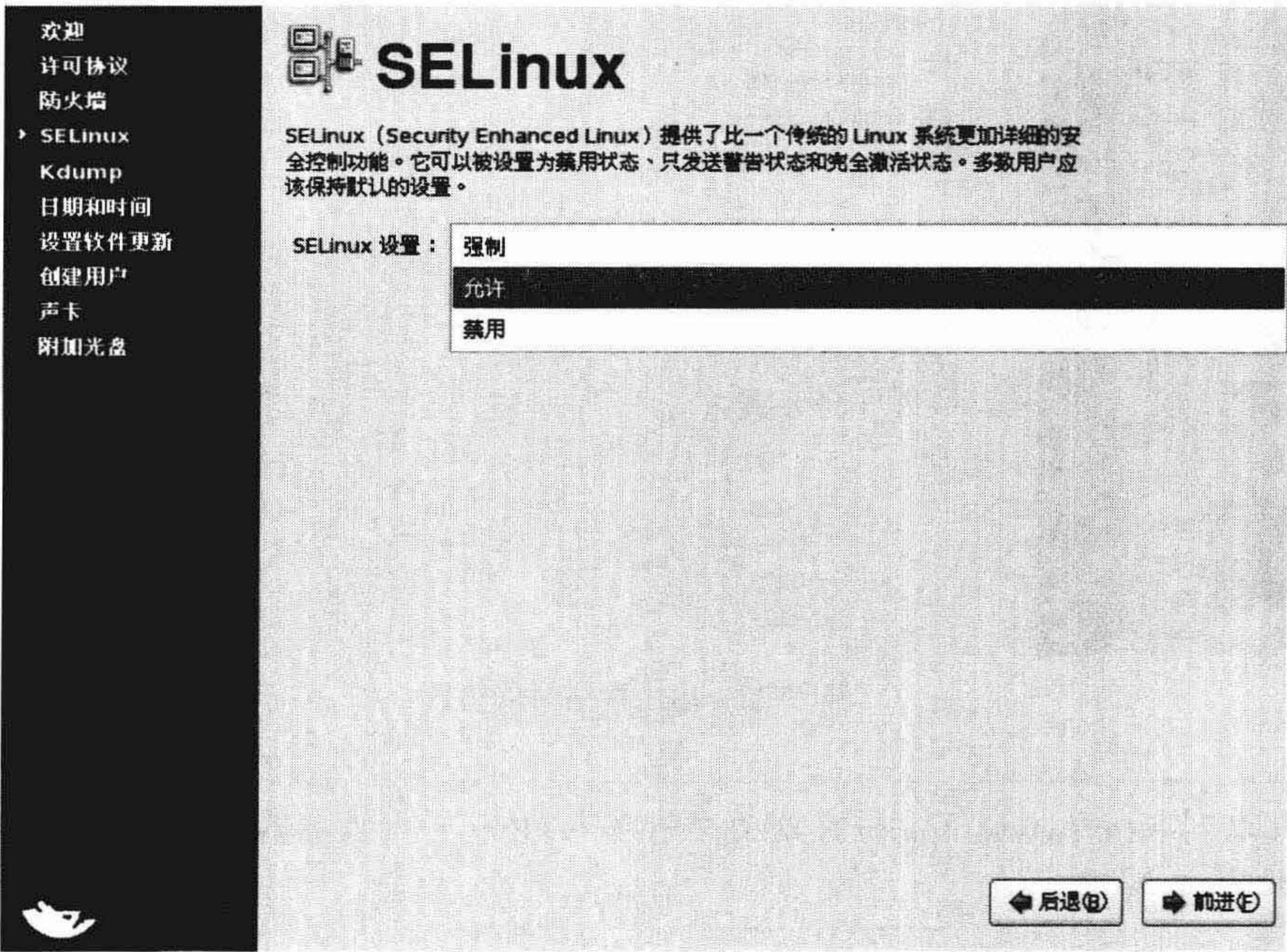


图 2-24 配置 SELinux

■注：主机不是必须使用 SELinux，如果不想用，你可以选择如图 2-24 所示的“禁用”关闭它。不是所有的 Red Hat 衍生发行版都默认运行 SELinux，因此不需要在所有的平台上都这么做。

在下一画面将要设置日期和时间，它通常用网络时间协议（Network Time Protocol, NTP）完成这件事。NTP 通过连接位于 Internet 上（所以需要先连接到 Internet）并与非常精确的时钟相连的时间服务器来设置时间。为了获得时间，主机会轮询这些时间服务器。它通常要轮询 3 个服务器（轮询的数量和服务器都可以配置），然后使用一个算法，根据响应和主机所在的时区决定最精确的时间。

在“网络时间协议”标签上选择“启用网络时间协议”选项，就启用了 NTP。Red Hat 有一些可用的时间服务器，并且它们会自动配置好。如果想添加自己的服务器，可以使用“添加”、“编辑”或者“删除”按钮。

“日期和时间”画面的高级选项区提供了两个选项，如图 2-25 所示：“开启服务之前先同步系统时钟”和“使用本地时间源”。如果硬件时钟不是非常可靠，请把本地时间源的功能去除。

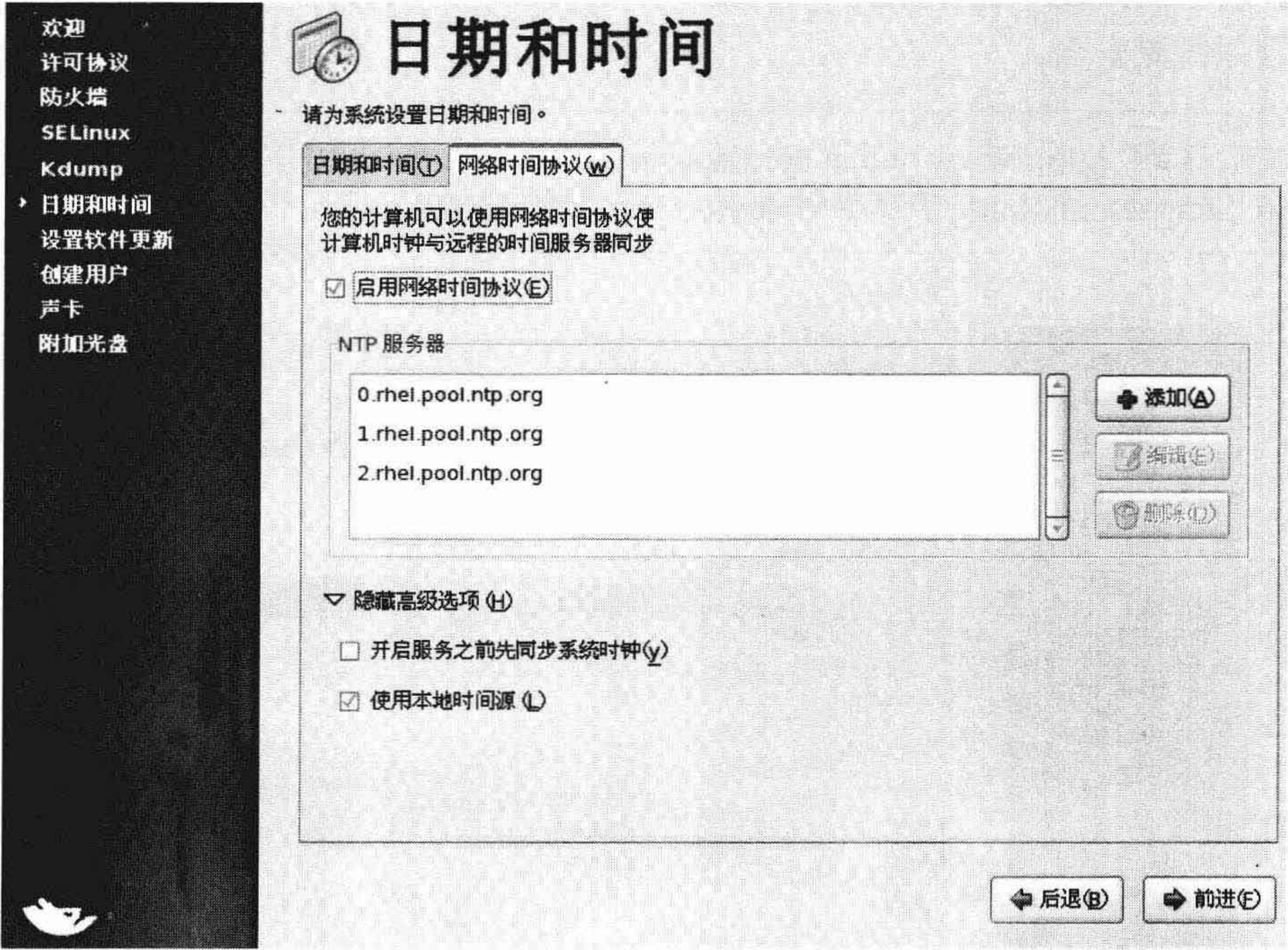


图 2-25 启用网络时间协议

■注：你可以到 <http://www.ntp.org> 了解更多有关 NTP 和时间服务器的内容。我们还会在第 9 章更详细地探讨 NTP。

安装进程尝试联系 NTP 服务器之后（假定你有一个 Internet 连接），会要求设置 Red Hat

提供的软件更新。这些更新只有通过 RHN 订购才可用（回想一下，RHN 是作为购买 Red Hat Enterprise Linux 许可证的一部分而接受的软件更新服务）。只有正确订购了它，该选项才有用。如果是这样的话，你就可以按照如图 2-26 那样选择“是，我现在注册”，并单击“前进”按钮。

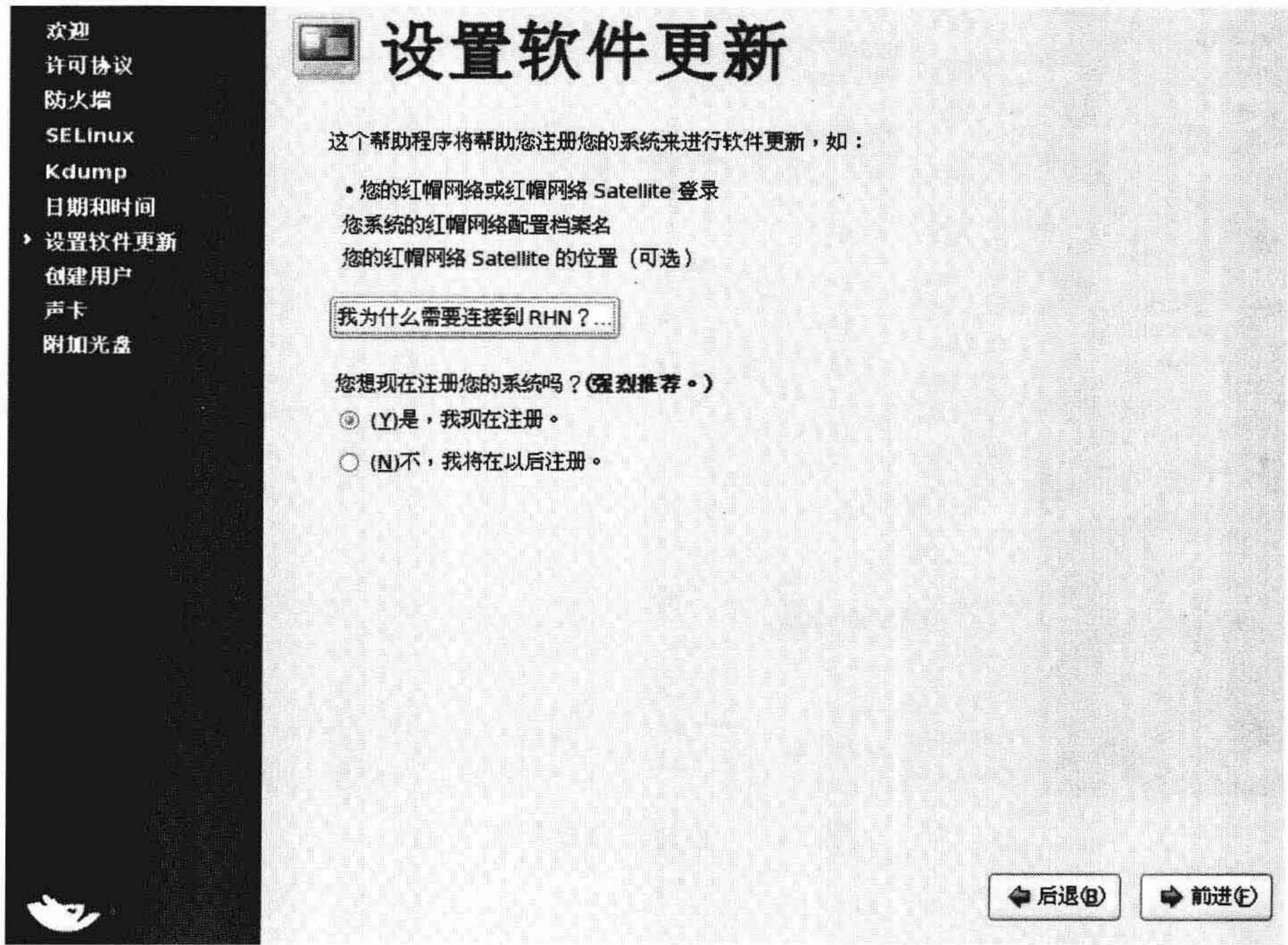


图 2-26 选择软件更新

接下来要选择获取更新的位置。可以通过 RHN 的门户或者通过 RHN 卫星服务器或代理服务器选择 Red Hat 提供的 Red Hat 网络。卫星和代理服务器是 Red Hat 的程序供应和管理组件的一部分。使用它们能够建立一个管理补丁和更新的分布式网络，只下载一次更新，然后分发给多个站点和主机。刚开始时你不太可能需要购买其中一个服务器，但是如果有多站点和大量主机，你也许就想研究它们的用途了；可以通过访问 http://www.redhat.com/red_hat_network/和 http://www.redhat.com/docs/manuals/satellite/Red_Hat_Network_Satellite-5.1.1/html/Proxy_Installation_Guide/s1-intro-proxy.html 来详细地了解这些产品。

提示：Red Hat 的卫星服务器产品有一个类似的免费产品，叫 Spacewalk(<https://fedorahosted.org/spacewalk/>)，它可用于如 Fedora Core 和 CentOS 之类的发行版。Spacewalk 是卫星服务器的一个上游开发版本，包含较新的功能，但比 RHN 卫星服务器更不稳定。

如图 2-27 所示，我们选择了默认选项，即直接从 RHN 得到更新。如果你的网络使用代理服务器连接到 Internet，那么可以单击“高级网络配置”按钮，在那里提供使用它所必需的资料。

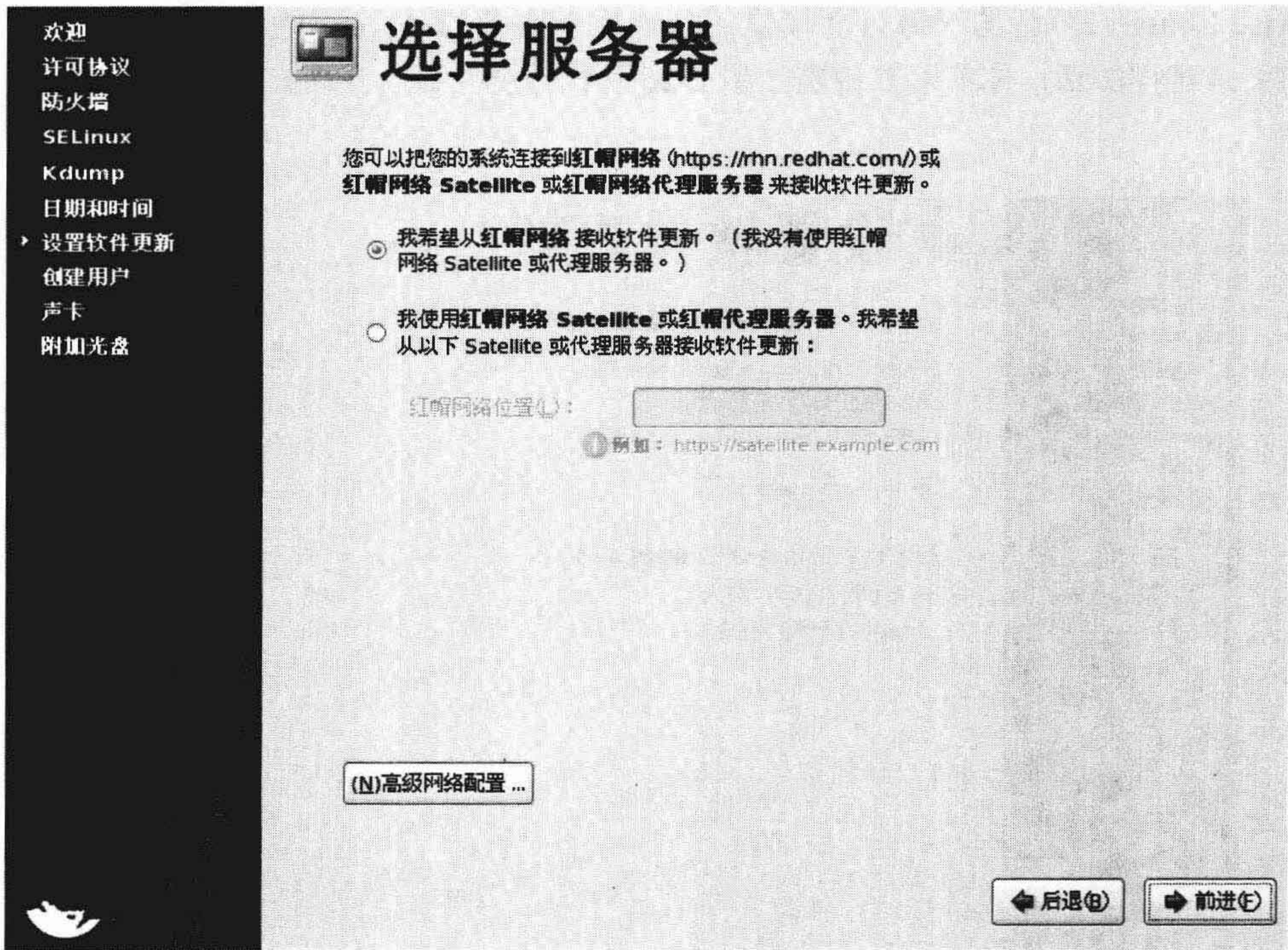


图 2-27 选择软件更新的来源

■提示：我们会在第 7 章更详细地探讨 RHN 和主机更新。你需要拥有 Red Hat 登录 ID、密码，并订购操作系统。如果没有的话，可以到 Red Hat 网站（<https://www.redhat.com/wapps/ugc/register.html>）上注册一个，还可以从 Red Hat 网站（<https://www.redhat.com/wapps/store/catalog.html>）在线购买。订购之后你可以从 Red Hat 下载安全补丁和其他可用的软件。

接下来要求输入 Red Hat 登录 ID 和密码，就是访问 RHN 网站时所使用的 ID 和密码。输入身份信息后，单击“前进”继续。登录过程和密码将被验证，如果两者都正确，就可以继续了。

然后要定义一个描述性的系统名（叫主机名更合适）以给 RHN 提供一个标识。如图 2-28 所示，输入主机名 `au-mel-rhel-1`。要确保选中“发送硬件配置档案”和“发送软件包配置档案”选项，以便充分利用订购服务，让 Red Hat 为系统提供所有更新。单击“前进”可以在给 RHN 发送这些信息之前复查资料。

现在有机会复查订购信息，如图 2-29 所示。如果没有订购的话，你将看到一个 Code 91 的错误信息。你可以核对 Red Hat 订购信息，如果需要，还可以为新主机额外订购一份。

接下来你将看到“完成更新设置”画面，它确认了主机在 Red Hat 的 RHN 上订购成功。单击“前进”继续。

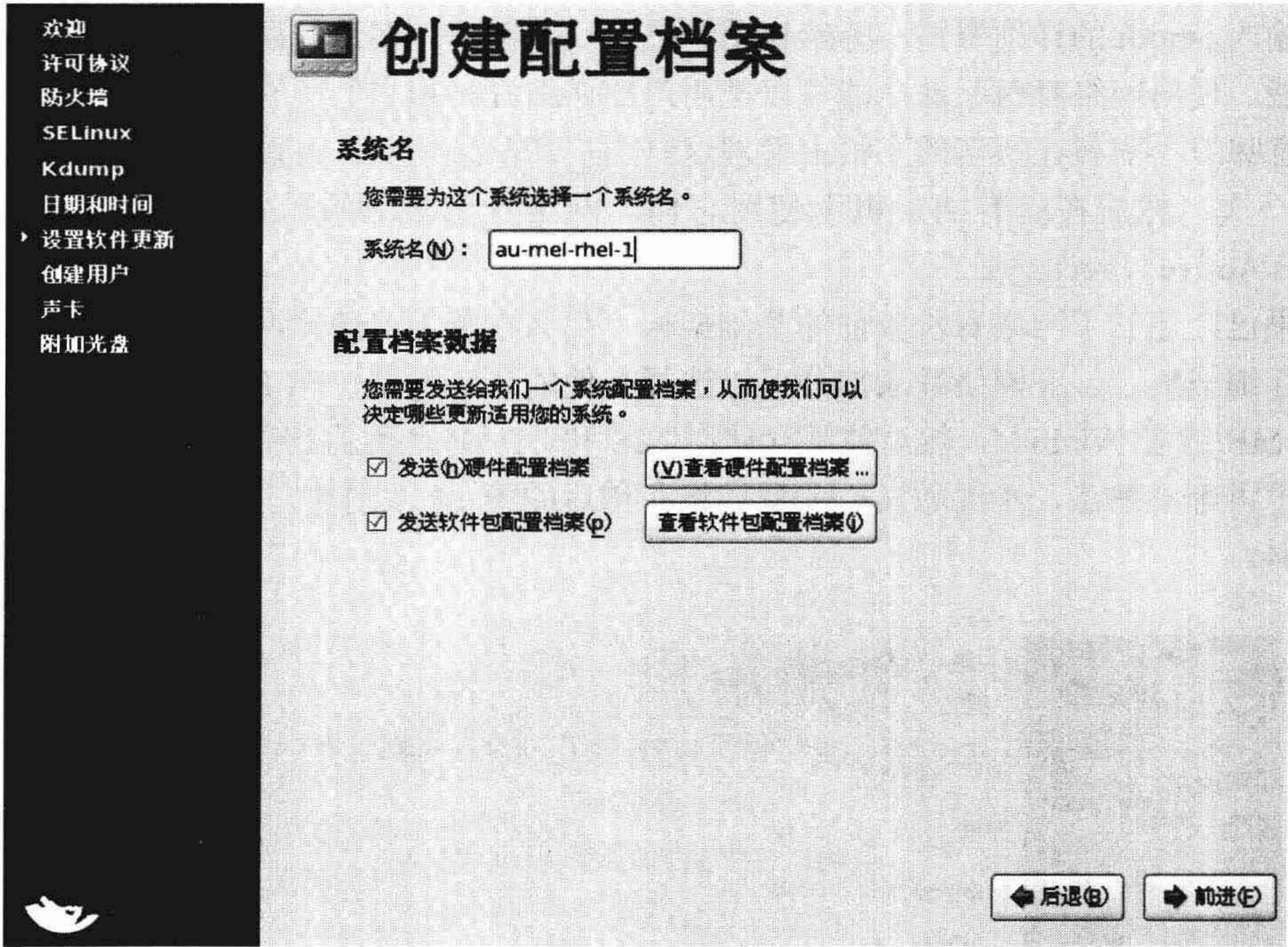


图 2-28 为 RHN 设置系统名

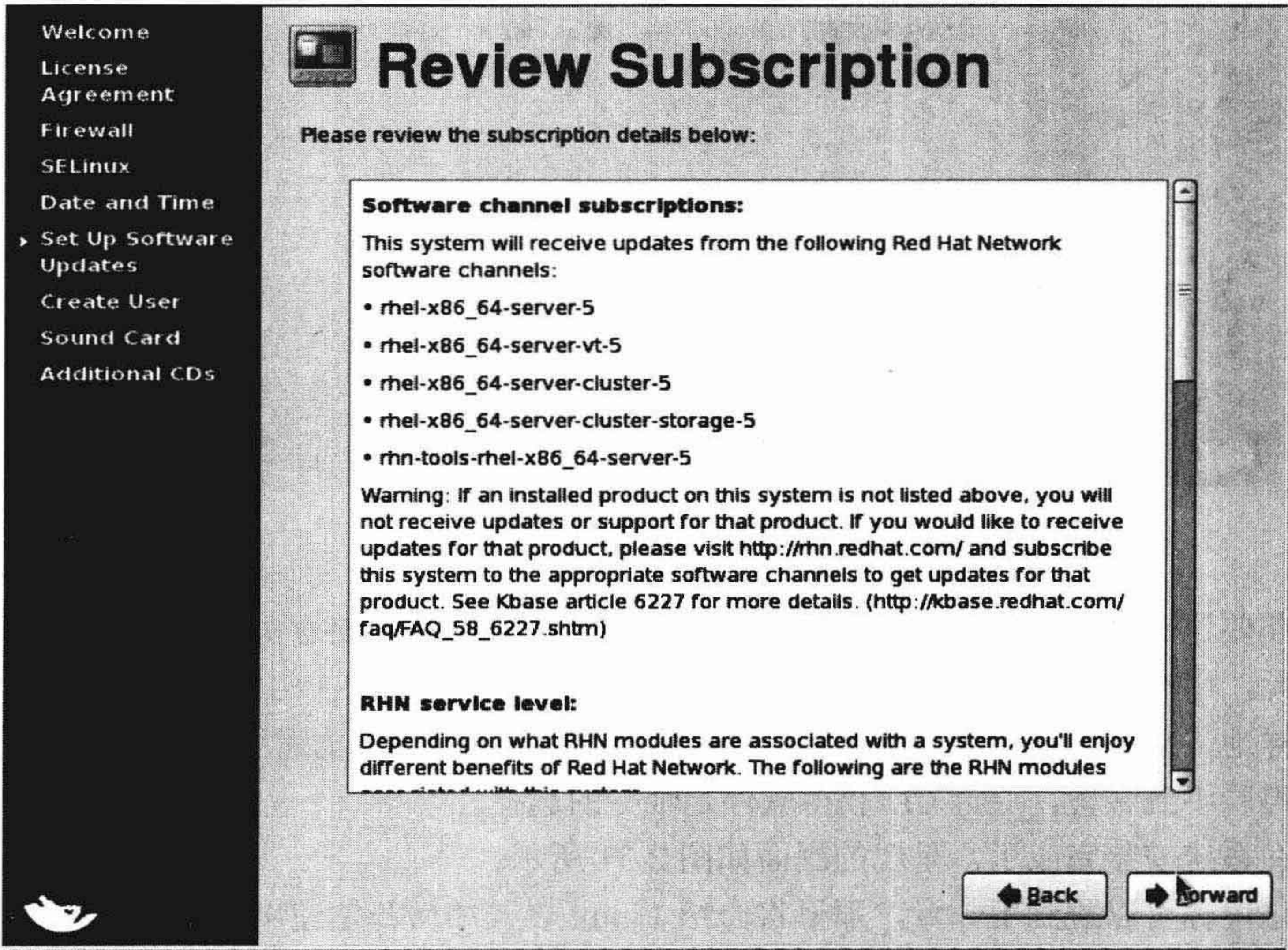


图 2-29 复查订购信息

设置好更新后，你现在要为主机添加一个新用户。在本例中我们将创建一个叫 jsmith 的用户。输入 jsmith 的详细资料，并给该用户提供一个密码。在这一画面还可以配置网络身份验证功能。使用网络身份验证功能可以把用户的详细资料和它们所有的权限存储在远程主机上。这意味着不需要在所有的主机上管理这些用户和信息凭证，可以把它们在这台远程主机上配置一次，然后在所有的主机上使用它们。网络身份验证的另外一个例子是 Microsoft Windows Active Directory。

如果已经建立了网络身份验证的基础架构，那么在单击“使用网络登录”按钮之后就可以输入详细资料了。在网络登录部分可以选择其他的认证类型，如 Kerberos、LDAP、Smart Card、SMB 或者 Winbind，然后使用存储在 NIS 和 LDAP 之类的数据库中的用户信息。如果对这些术语都不熟悉，不要恐慌——我们将在稍后的第 16 章阐述它们。现在使用图 2-30 所示的资料。

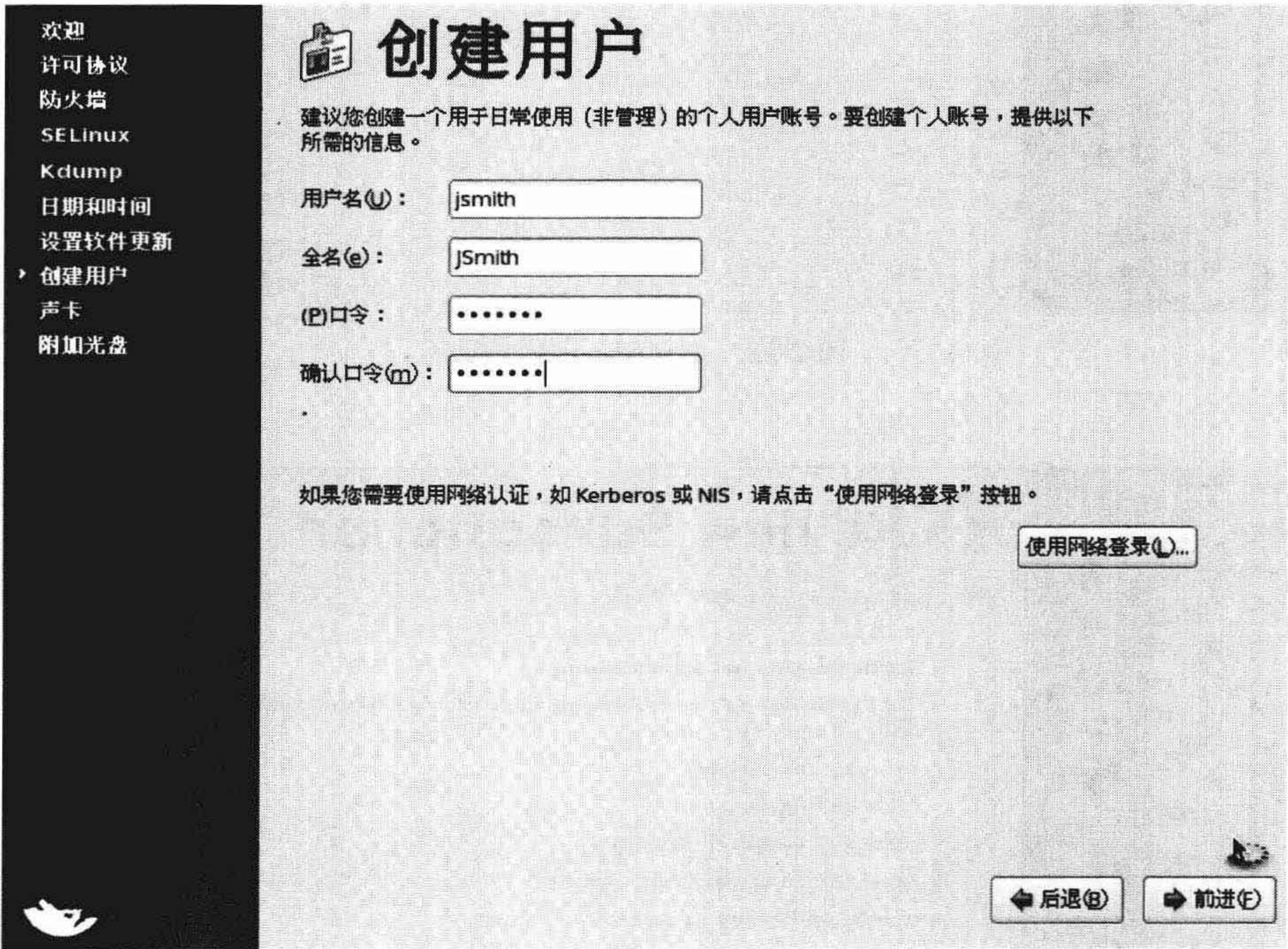


图 2-30 创建第一个用户

除非要建立桌面系统，否则我们不推荐安装声卡。这是历史遗留的问题，那时声卡价值数百美元，用在新系统上被认为华而不实。而如今，声卡在主板上免费赠送。

同样地，你也可以忽略附加 CD 的部分。这差不多也是历史遗留的问题，那时 Internet 连接慢得看不到希望，通过 CD 获得软件是唯一的选择。

现在系统安装完成了，呈现的画面如图 2-31 所示。

从这里你可以跳到下一章，那里将介绍 Linux 以及如何开始使用新主机。另一个备选项则是继续阅读本章，学习如何安装 Ubuntu。

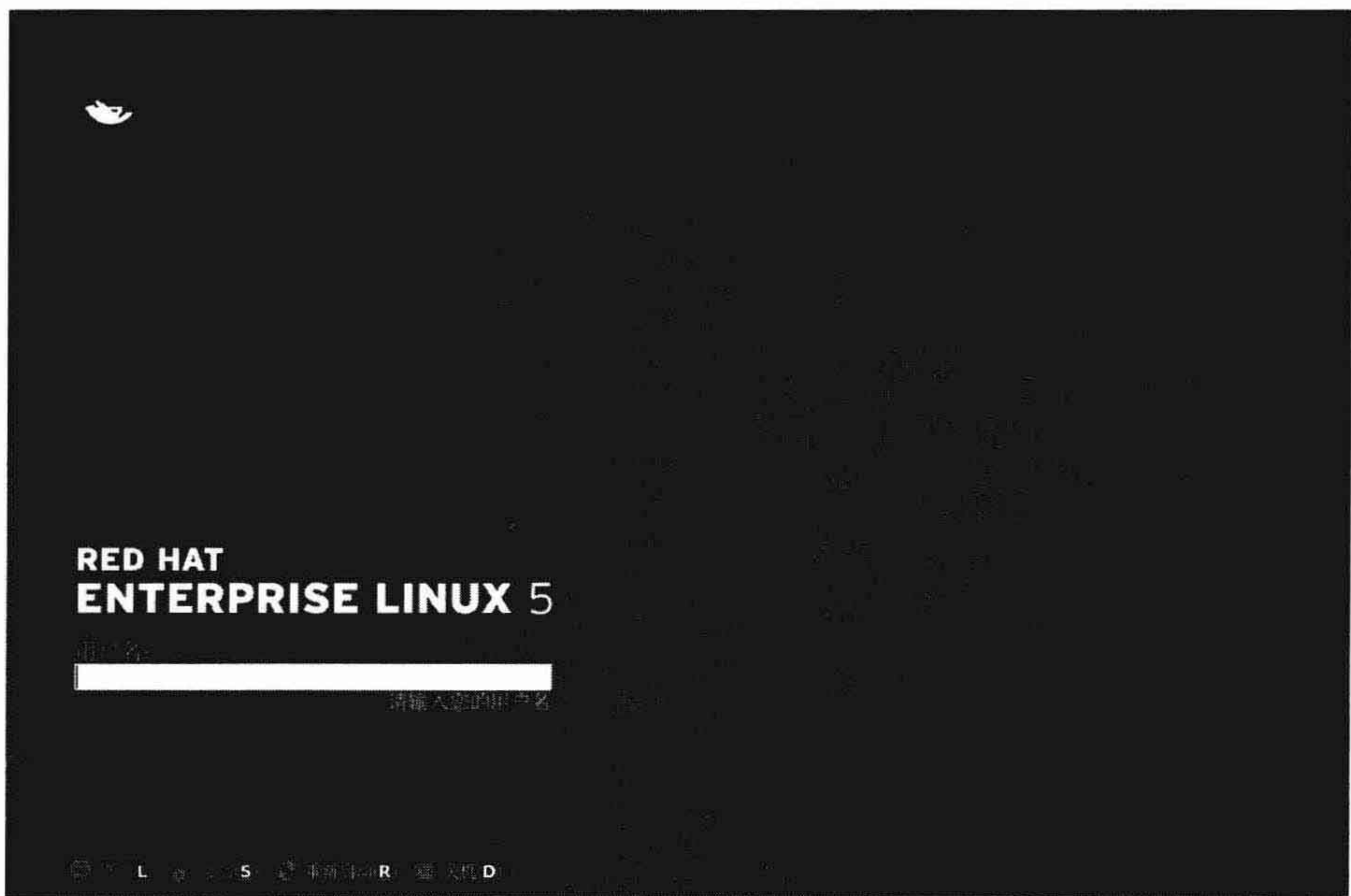


图 2-31 安装完成，系统已可用

2.3 Ubuntu 的安装

Ubuntu 有两种：桌面和服务端。桌面版本的设计目的是配置为桌面系统，而服务器版本是为服务器主机设计的。本节将演示如何安装服务器版本。安装 Ubuntu 服务器和安装 Red Hat Enterprise Linux 服务器的过程非常相似。主要的概念是相同的：选择语言和键盘布局，选择想要的磁盘分区方式，然后选择想要安装的软件包。

为安装 Ubuntu，我们需从 Ubuntu 网站下载一个 ISO 文件，它包含了完成安装所需的很多数据。这次练习将使用来自 <http://www.ubuntu.com/getubuntu/download> 的完整 CD。

■注：Ubuntu 和 Debian 合理地利用了网络安装程序（net installer），使安装更灵活。网络安装程序是操作系统的小型版本，通常以 ISO 文件的形式提供。该文件可以烧录到 CD 中，然后从 CD 启动。网络安装程序为主机提供启动和开始安装程序所需的基础，然后所有附加的软件或应用程序从在线软件资料库下载。这意味着安装新主机需要连接到 Internet。这还意味着通过标准的 ADSL2 安装一个完整的 4GB 的操作系统会花费相当长的时间，但是使用网络安装程序对于加载较小系统是一种很好的方式。我们将在第 19 章研究预备多个系统的方法时深入探究网络安装程序。

这里做以下几个假定。

- 所使用的 ISO 是 Ubuntu 网站（<http://www.ubuntu.com/getubuntu/download>）上的 Ubuntu 8.04 LTS 服务器版本，并且已经把它烧录到 CD 中。在下面这个网站可以找到更多关于如何把 ISO 文件烧录到 CD/DVD 的信息：<https://help.ubuntu.com/>

community/BurningIsoHowto。

- 所构建的只是一个包含基本的邮件、DNS 和网络功能的服务器，与在 RHEL 安装时所做的一样。
- 安装所用的服务器是新的，先前没有任何操作系统。

把 CD 放到 CD 驱动器中并打开主机电源，然后就是选择安装要用的语言。如图 2-32 所示，我们选择了 English。

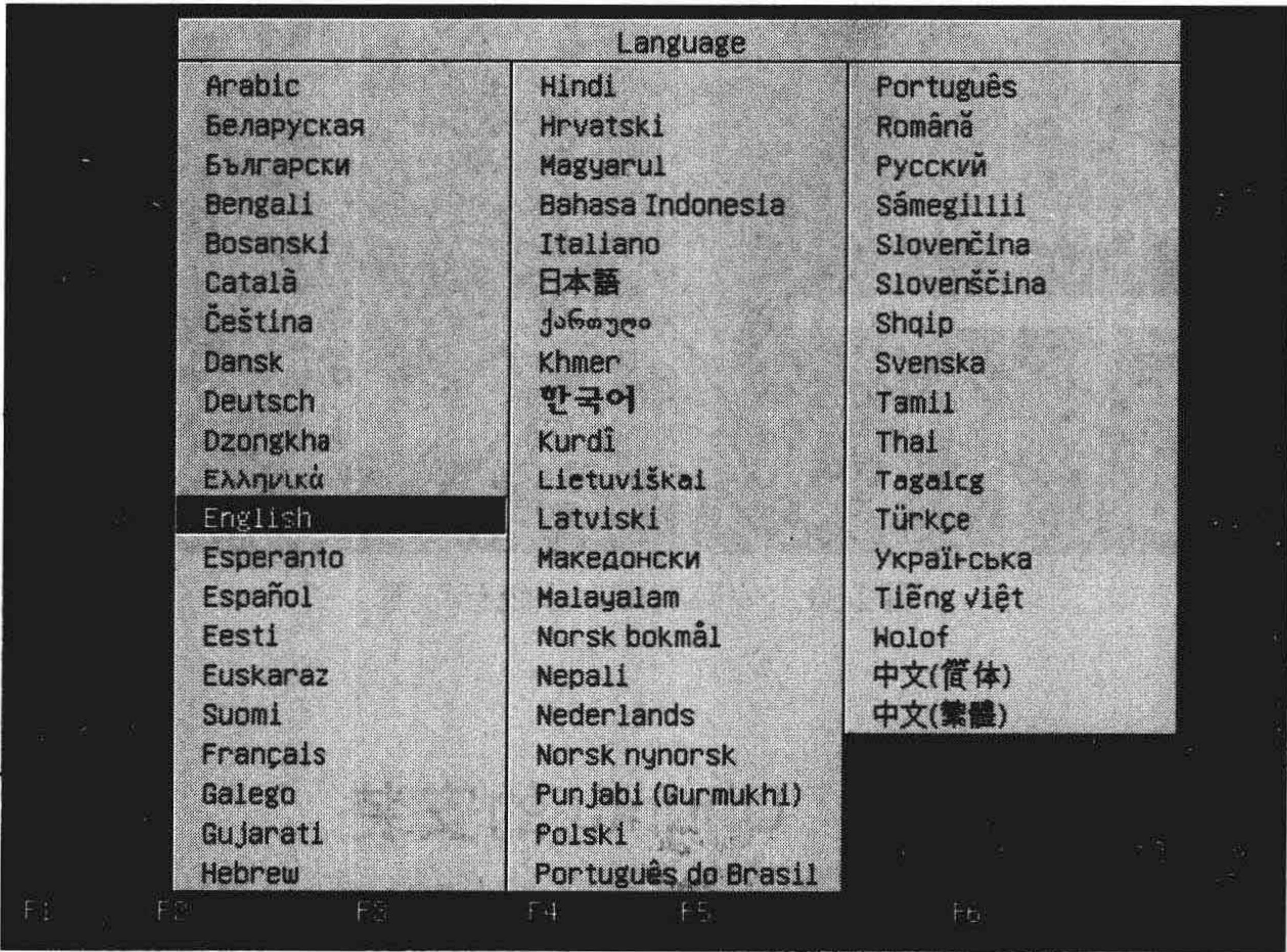


图 2-32 选择安装语言

接下来一个画面（见图 2-33）给出了一组类似于 RHEL 安装时出现过的选项。F1 提供一个全面的帮助系统，它解释如何使用安装程序以及如何处理特定的硬件。F2 和 F3 允许修改语言和键盘映射。F4 在服务器磁盘上不使用，但是在桌面安装程序磁盘上可用来选择不同的安装类型。F5 允许把显示器更改为高对比度模式以及初始化屏幕阅读器或者点字显示器。最后，F6 提供手工编辑启动命令的选项。

在这一画面还可以测试 CD 是否有故障，然后进入挽救模式，这些会在第 8 章讲述。

Test memory 选项其实不引导 Linux，而是启动一个叫作 memtest86 的实用工具。这个实用工具反复把整块的数据写入 RAM，然后读回，看看内容是否已改变。如果遇到随机的系统故障或者系统死锁，首先应做的事情之一就是测试 RAM 以确保它安置合适而且没有损坏。

如果选择 “Boot from first hard disk”，系统就会读主引导记录，然后就好像没有 CD 存在那样启动系统。第 3 章将更详细地讲解启动过程。

按回车键选择 “Install Ubuntu Server” 选项，然后安装就会开始。

接下来就是安装中的第一个选择：整个安装过程中希望使用的语言（见图 2-34）。这也会是系统最终的默认语言。



图 2-33 Ubuntu 服务器闪屏画面

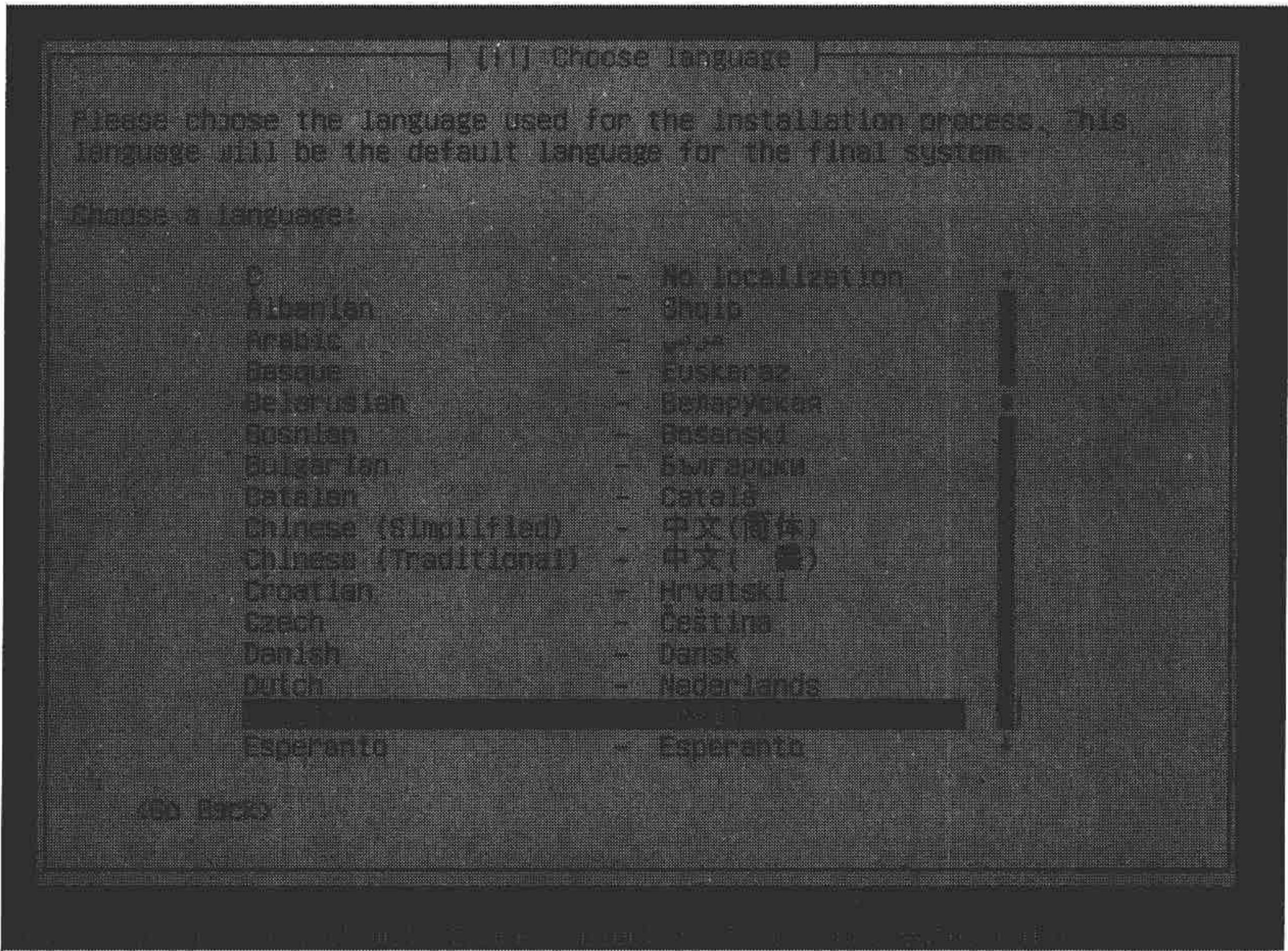


图 2-34 选择安装过程中的语言

然后要选择区域。这是指正在安装的服务器所在的地理位置。图 2-35 中选择了澳大利亚。

接下来选择键盘和键盘布局。如前所述，键盘布局就是正在使用的键盘映射。不同的地区会有不同的映射，所以选择最适合你的区域和语言的键盘布局。如图 2-36 所示，在这里选择 “Yes” 将引出一系列深入的问答，通过这些问答，Ubuntu 让你按不同的键，以试图找出所使用的键盘类型。

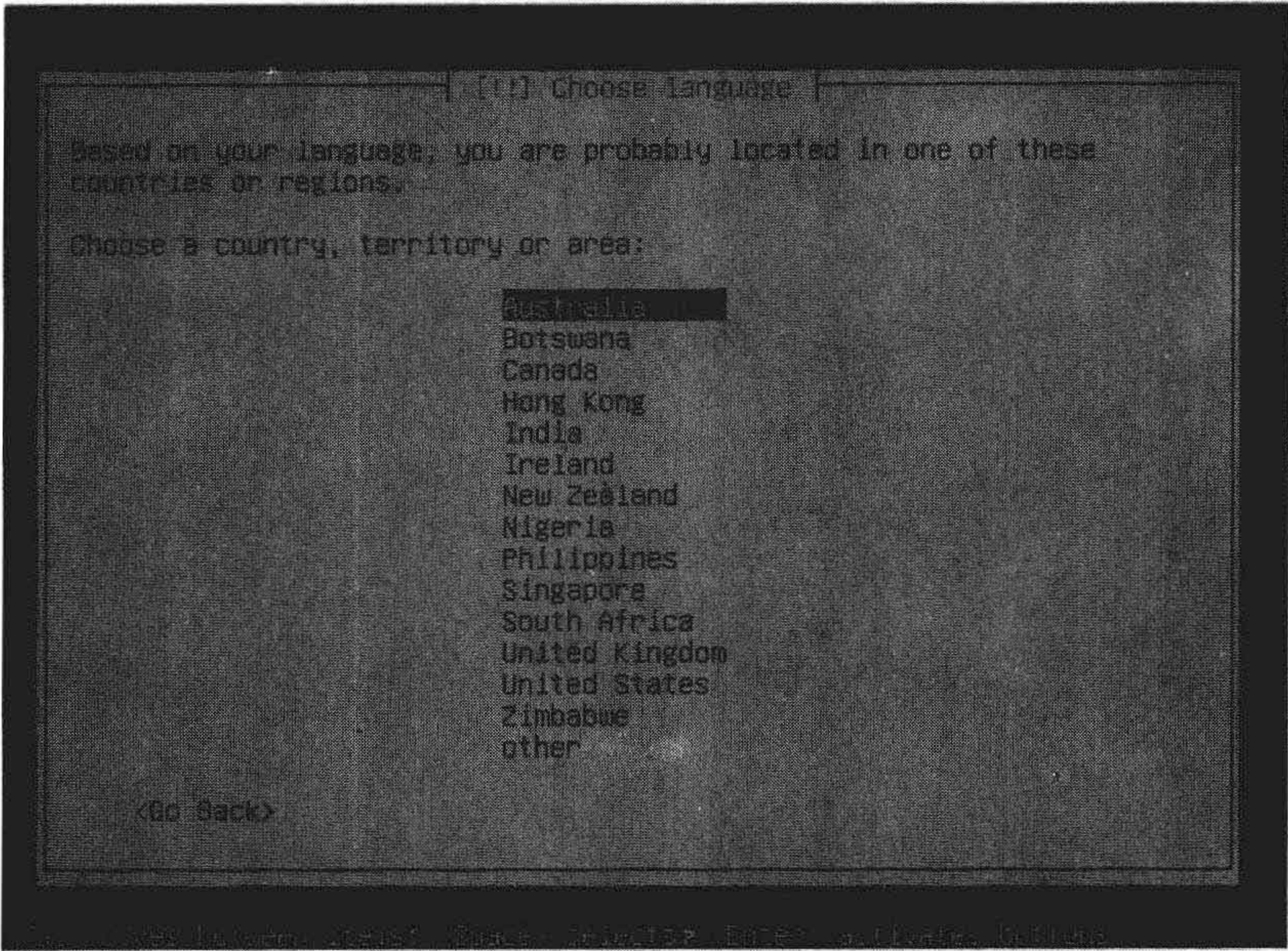


图 2-35 选择区域

选择 “No” 可节省时间，然后直接告诉安装程序你正使用的键盘类型。这里的默认选项对大多数的安装都能工作，不过你可以自由选择最适合你所在的地区的那一个。从图 2-36 开始是一系列展示键盘选择过程的屏幕截图。

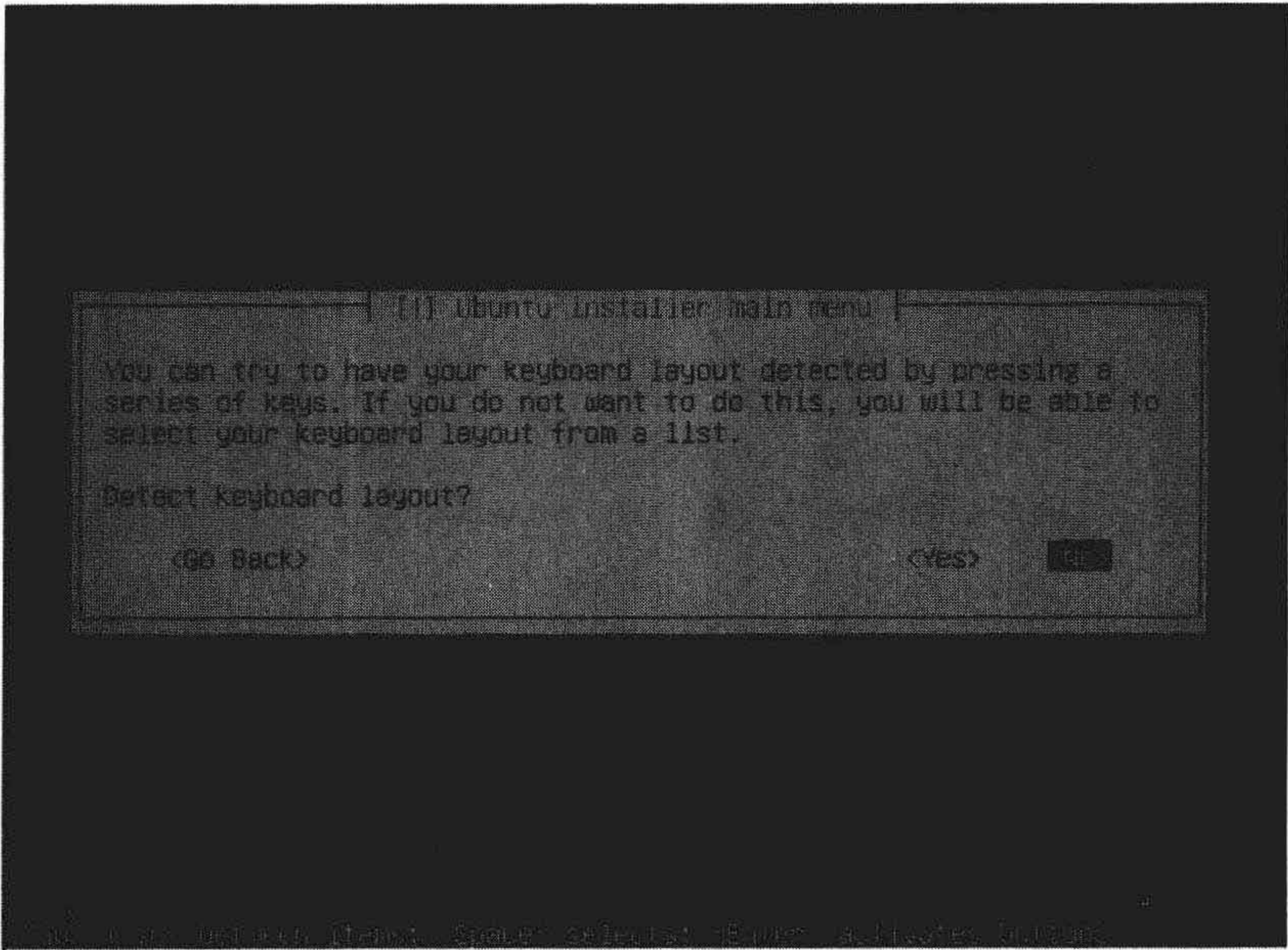


图 2-36 Ubuntu 试图检测键盘

选择 “No” 之后，图 2-37 所示的画面出现了，这里要选择键盘的产地。我们选择 USA，然后继续。

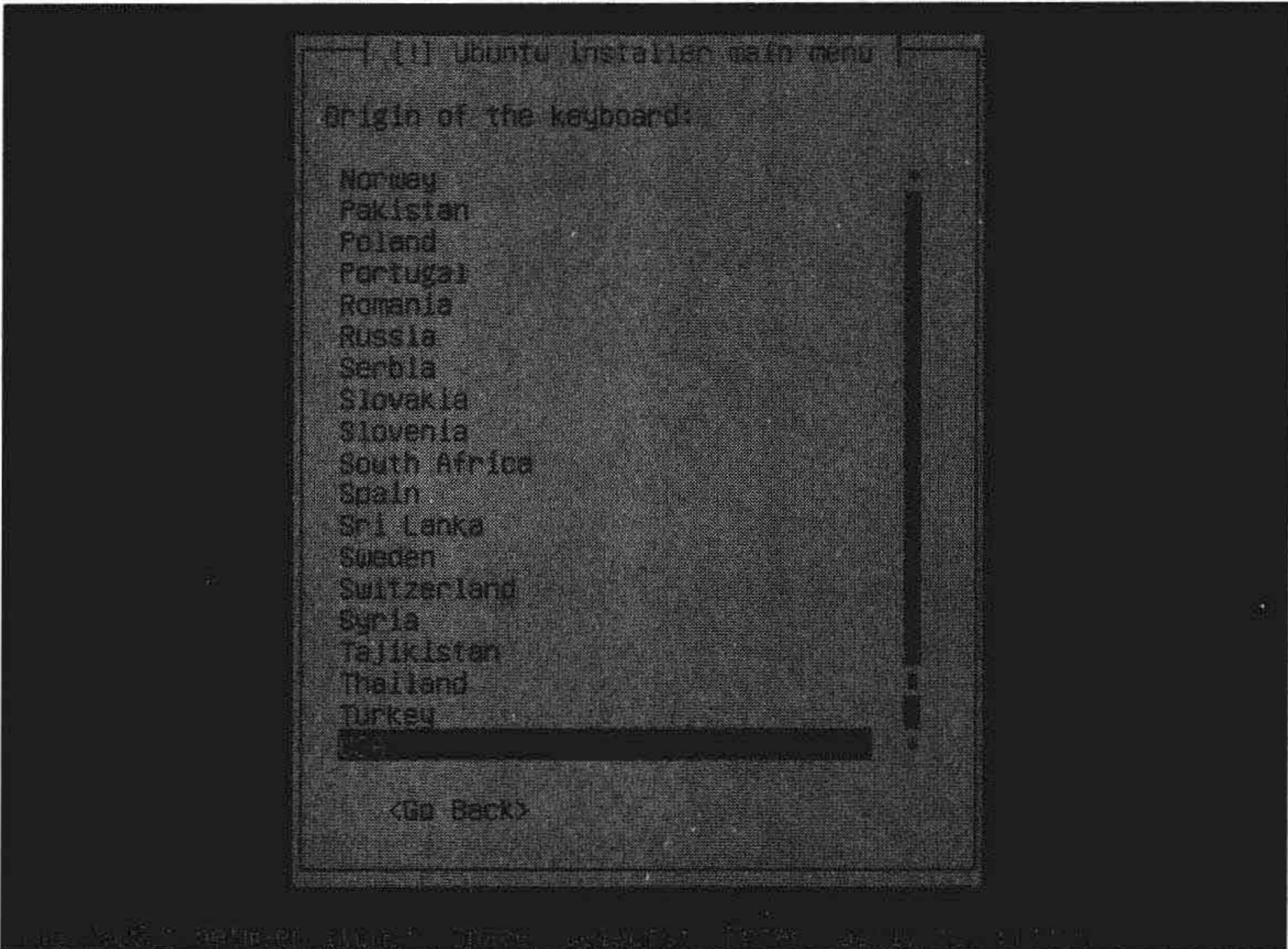


图 2-37 选择键盘的产地

图 2-38 选择了 USA 的键盘布局，它提供了澳大利亚计算机的标准键盘映射。

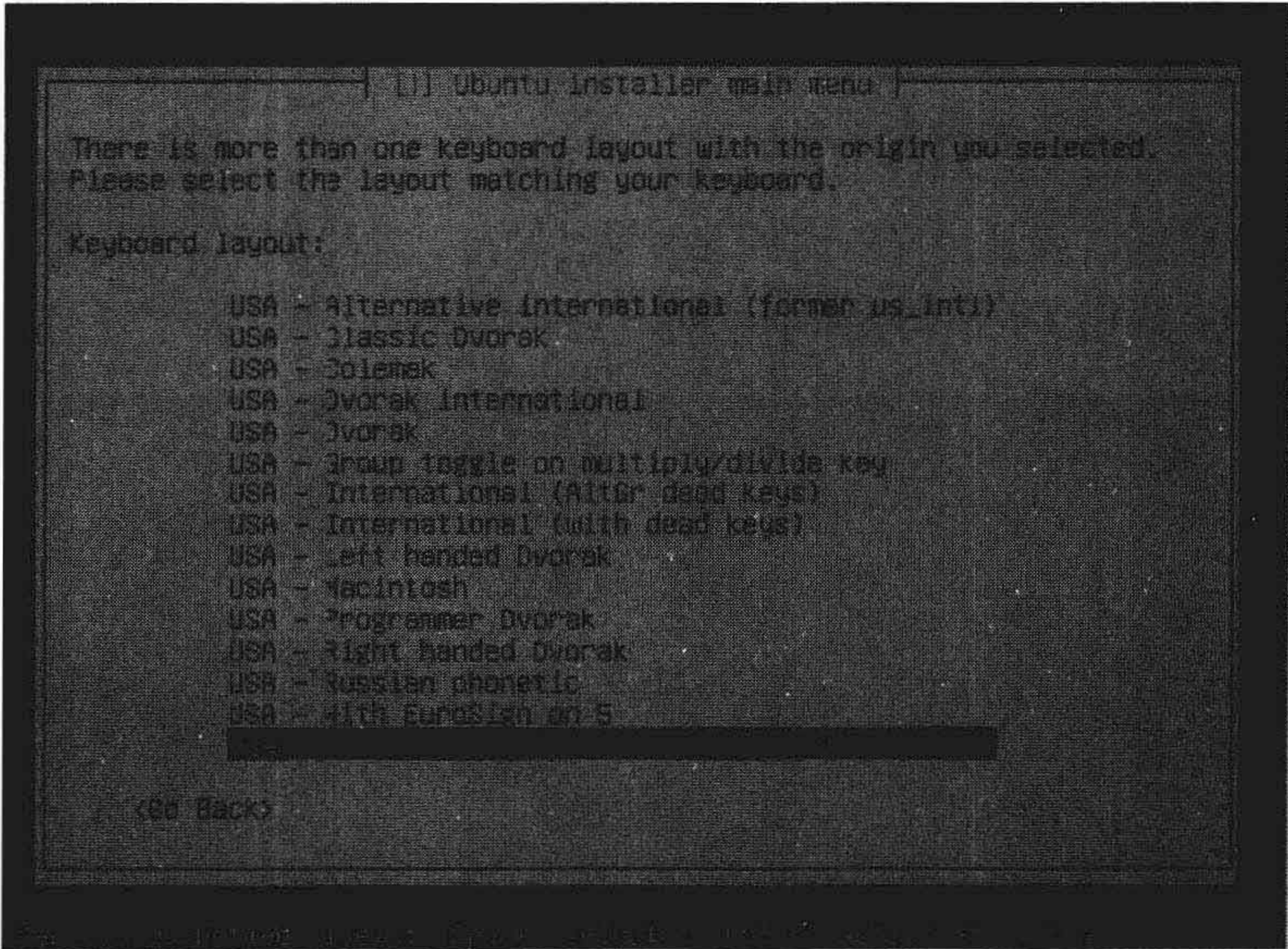


图 2-38 选择键盘布局

■提示：你可以在安装完成后随时改变键盘设置。

Ubuntu 安装现在做一次暂停以探测硬件并获得目标主机的更多信息。这个过程完成后，它会提示提供新主机的主机名。如图 2-39 所示，我们在这里输入 au-mel-ubuntu-1，因为它符合在 RHEL 安装一节中所讨论的命名标准。

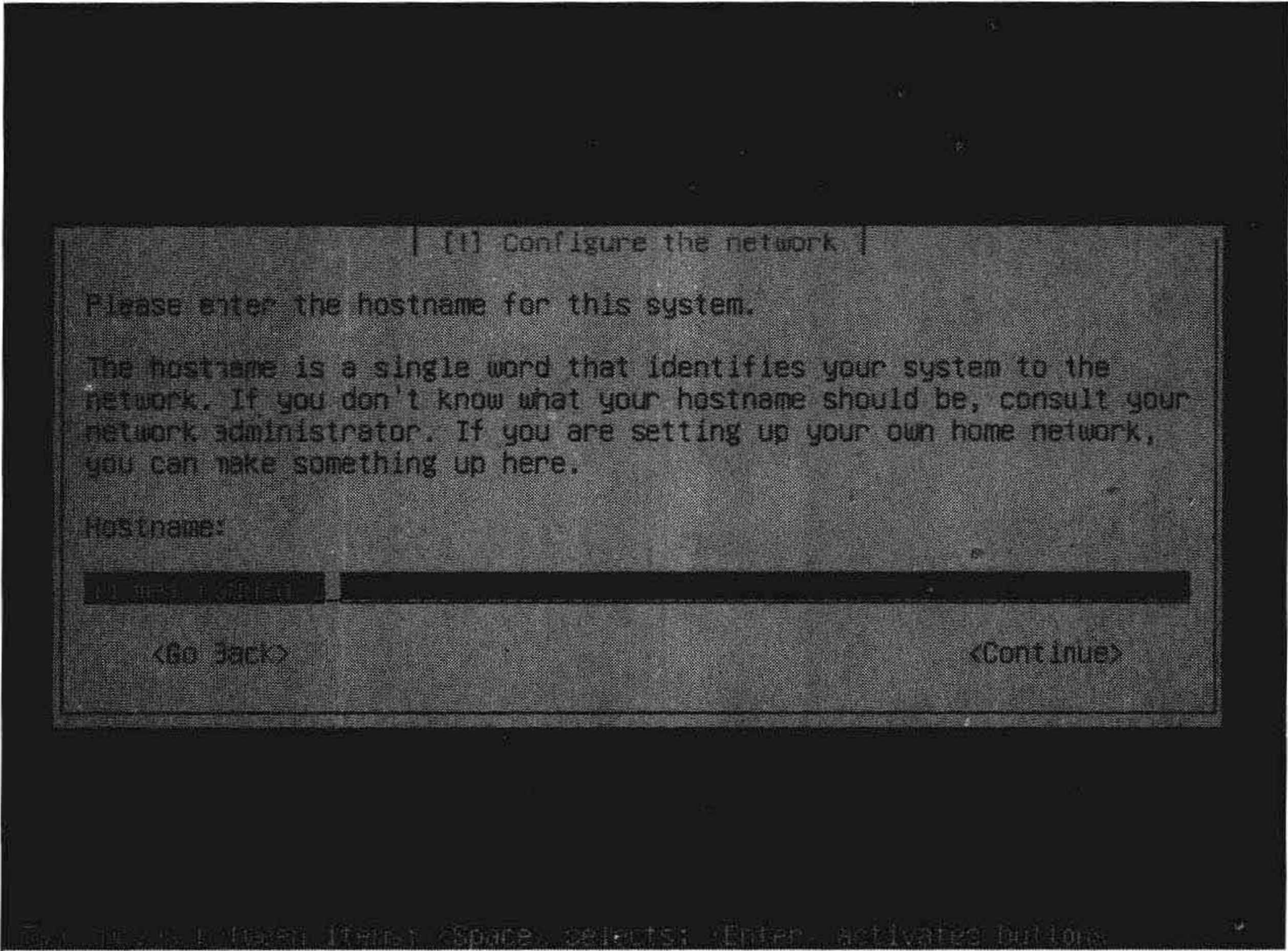


图 2-39 设置主机名

■提示：如果你在用网络安装的 ISO，它会要求检索你所希望安装的应用程序的位置。这些位置叫作归档镜像（archive mirrors），是 Ubuntu 软件的在线资料库。选择离你最近的地理区域；以我们为例，我们会选择澳大利亚。

接下来要指定时区，就是选择离你最近的省会城市。这样就可以使用正确的时区信息设置主机的内部时钟。我们选择 Melbourne，如图 2-40 所示。

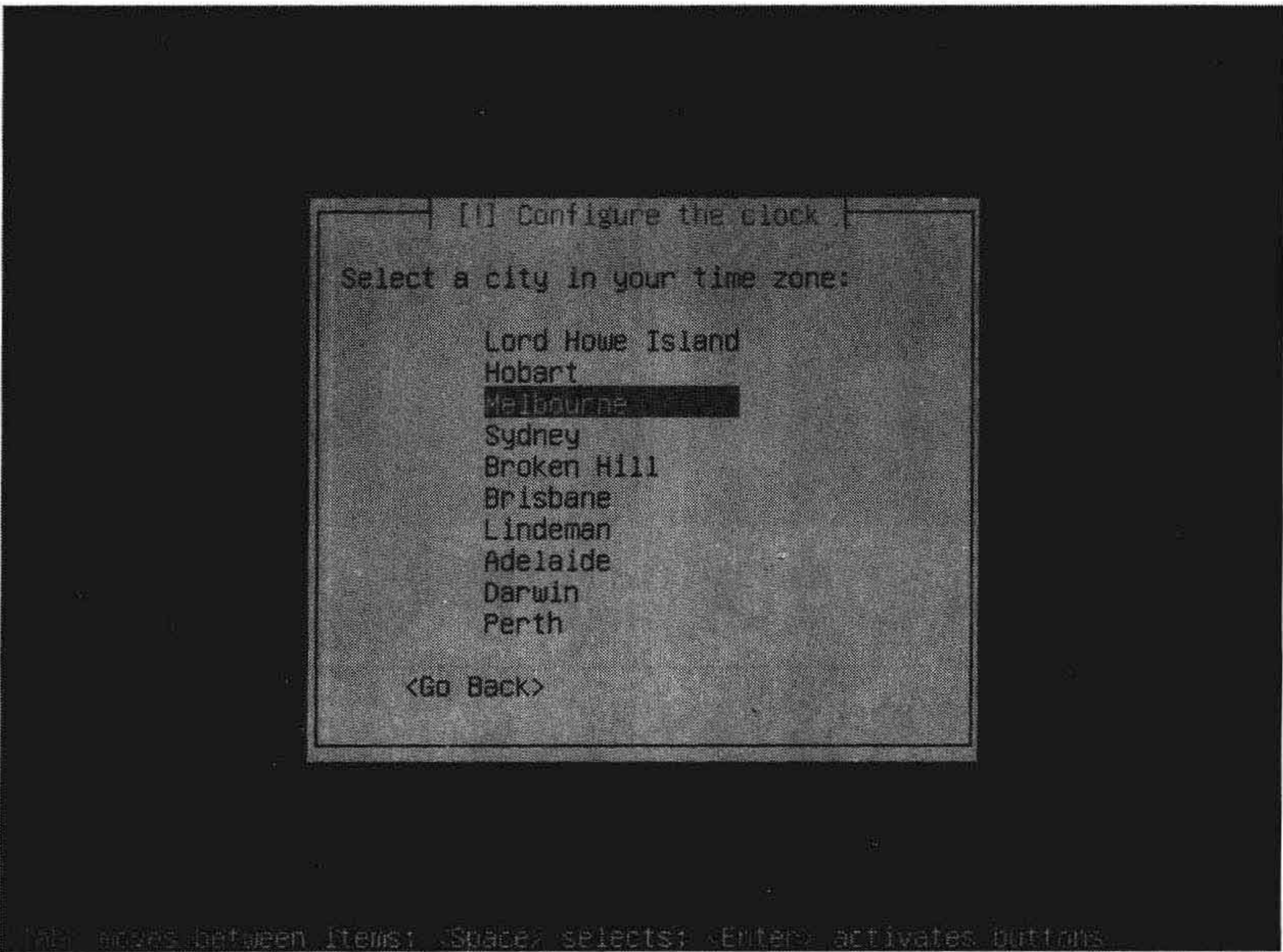


图 2-40 选择时区

接下来要为主机分区。它与安装 RHEL 时所描述的分区相同。你可以根据主机的需求把磁盘划分为不同大小的分区。同样，与 RHEL 安装时一样，它会提示你从几种默认的分区分项中选择其一或者使用分区工具定制自己的分区。

- **Guided – use entire disk:** 该选项要求选择一个硬盘。系统会把它完全擦除，然后创建一个 root 分区和一个 swap 分区。
- **Guided – use entire disk and set up LVM:** 该选项也擦除所有数据。然后它创建一个小的 boot 分区，并使用剩下的磁盘分配为 LVM 的 root 卷和 swap 卷。
- **Guided – use entire disk and set up encrypted LVM:** 该选项和前一个选项相同，只是所有 LVM 数据被加密。它要求提供一个密码。注意，在启动时需输入这个密码，因此该选项不适合远程或无头服务器 (headless server)。如果加密密码丢了，将不能检索数据。
- **Manual:** 该选项打开分区编辑器，允许手工配置分区、软件 raid、加密方法和 LVM。如果先前已经存在一个 Windows 系统并希望调整其大小，那么就应选择该选项。

对于范例主机，我们希望使用“Guided – use entire disk and set up LVM”选项。这样就使用了整个磁盘，还利用了逻辑卷管理 (LVM)。正如在 RHEL 安装那一节中所描述的那样，LVM 是一种强大的分区和磁盘管理方式，它为以后修改分区布局提供了更大灵活性。

■注：我们会在第 8 章详细讲述 LVM。

图 2-41 显示了默认的分区分项。

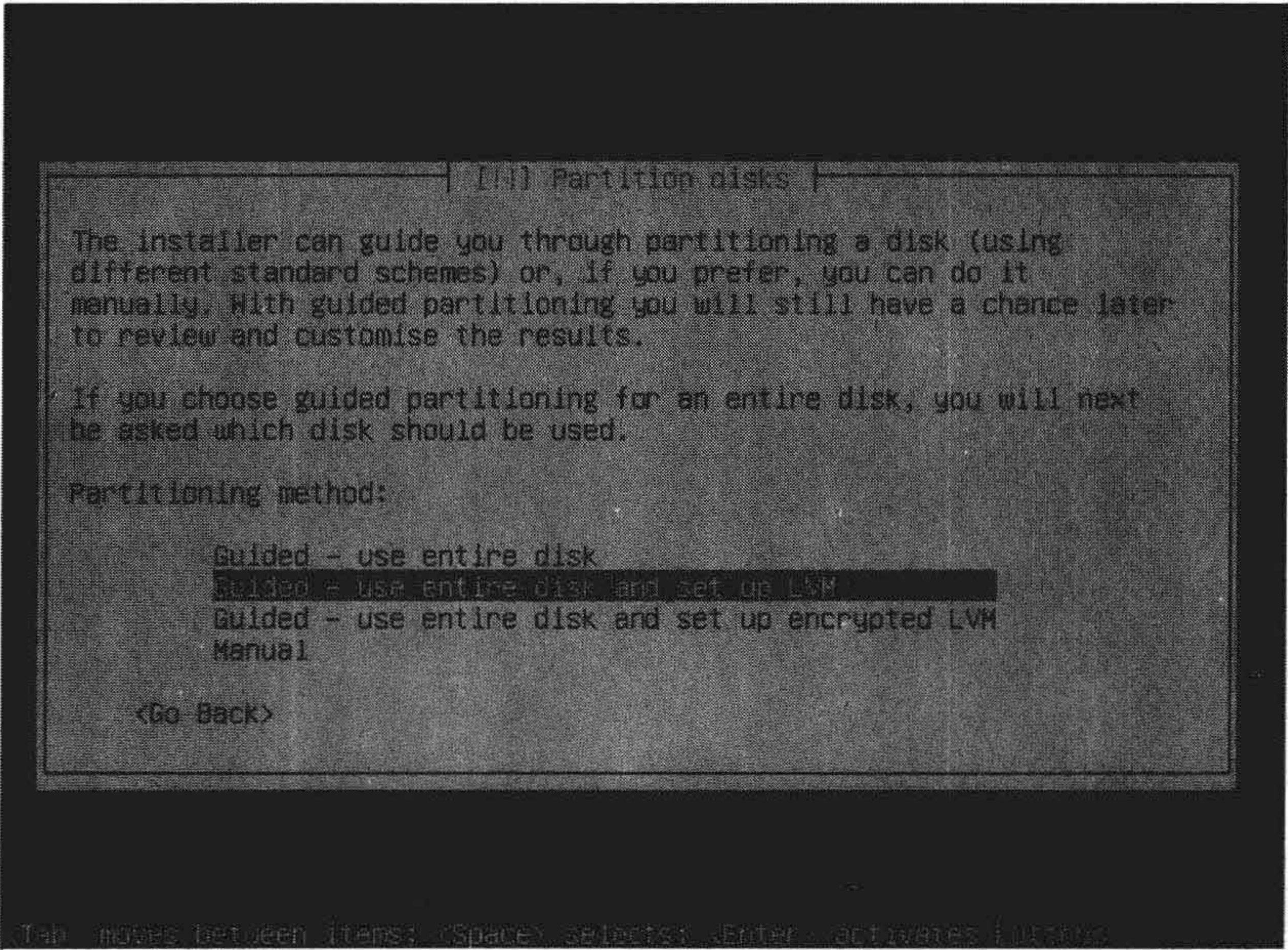


图 2-41 选择 LVM 对磁盘分区

接下来，选择希望在其上执行分区任务的驱动器。我们只有一个磁盘可供选择，如图 2-42 所示。

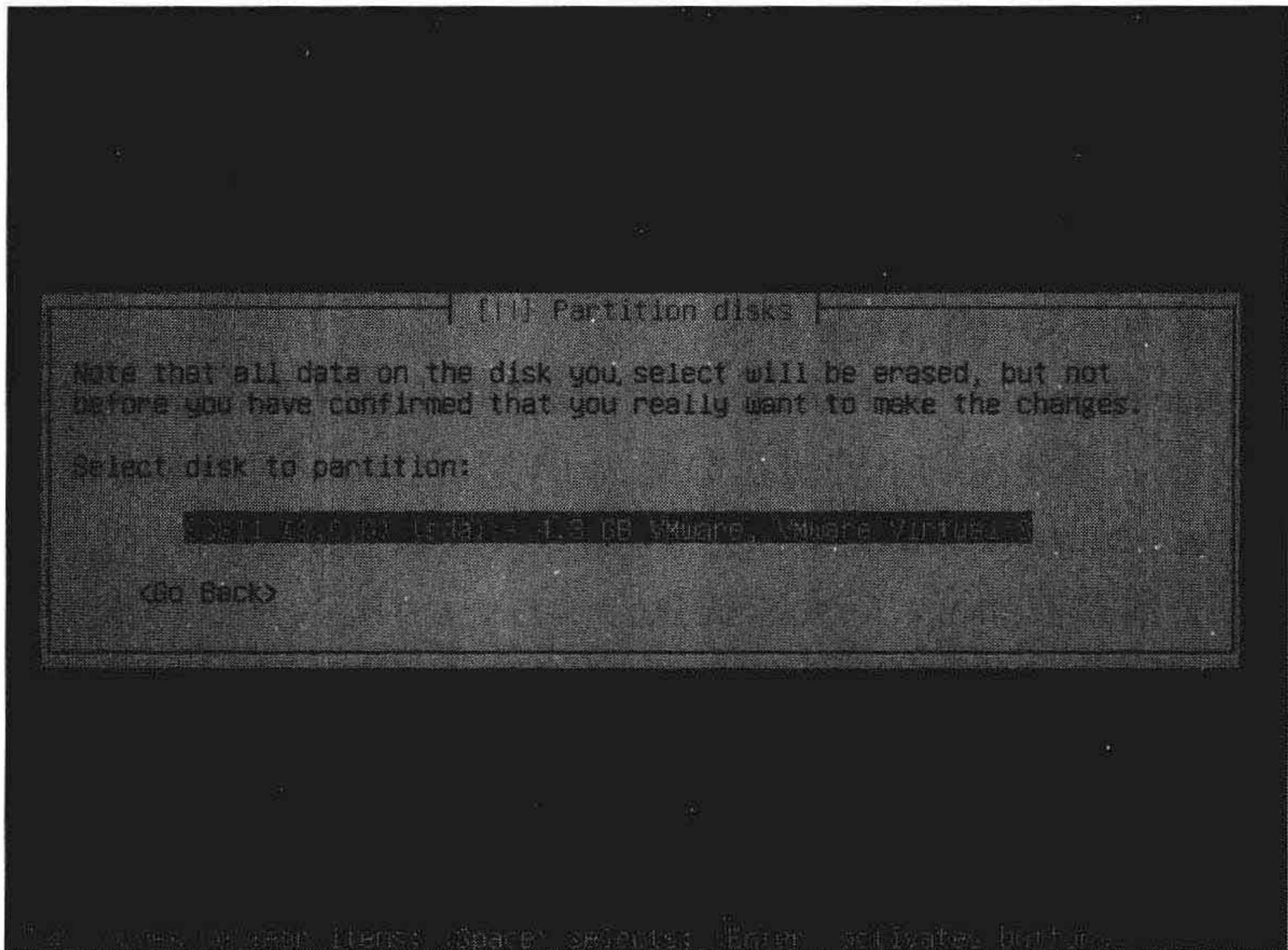


图 2-42 选择要分区的磁盘

■注：如果磁盘已含有分区，它会提示你是否覆盖它们。如果确信要这样做，就选“**Yes**”继续。如果正在以前的系统上安装它，在这里选择“**Yes**”将可能毁掉你拥有的所有已存数据。如果不确信，那么选择“**No**”。可代替它的方法包括使用像 PartitionMagic (<http://www.symantec.com/norton/partitionmagic>) 之类的工具对主机重新分区，在已经没有数据的硬盘上安装，或者在虚拟机上安装。

下一画面（见图 2-43）确认是否想往选定的磁盘写分区信息。分区信息需要先写到磁盘，然后才可以配置 LVM。选择“**Yes**”进入下一画面。

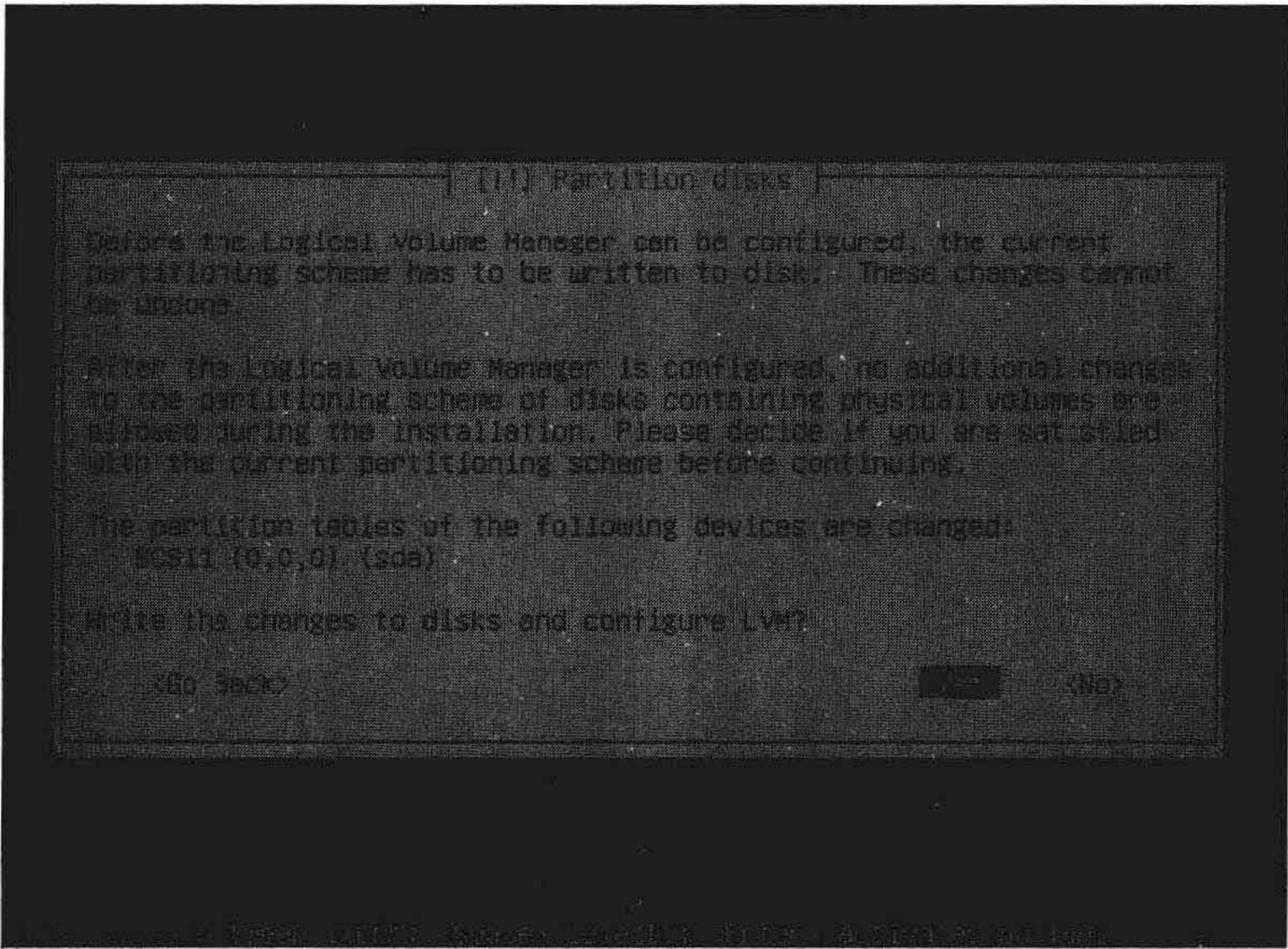


图 2-43 把分区信息写入磁盘

现在显示的是 LVM 分区布局，它显示少量的交换空间和剩余的 root 或/分区。当选择“**Yes**”

确认这种布局后，图 2-44 所示的 LVM 分区就完成创建并格式化了。

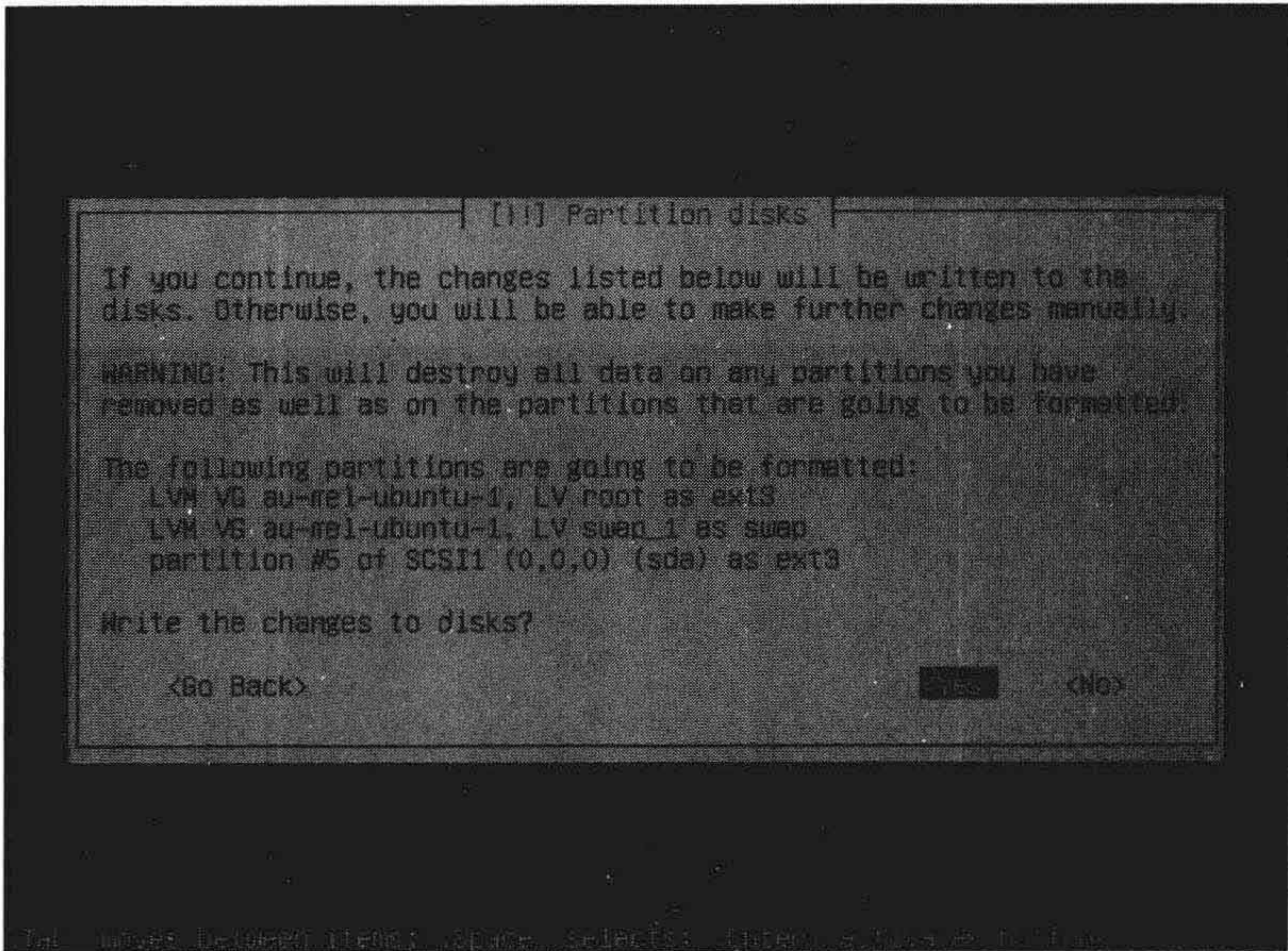


图 2-44 把 LVM 分区改动写入磁盘

■注：交换空间是硬盘驱动器上附加的存储，用来存储从 RAM 中溢出的数据。如果发现主机经常性地使用全部的交换空间或者频繁地交换，那么就可能需要调整主机，通常就是增加 RAM。第 8 章和第 17 章将更详细地阐述交换空间。

在安装过程的这个阶段，Ubuntu 开始安装必需的基本软件包，然后才能安装操作系统的余下部分。如果你正使用网络安装，根据 Internet 连接情况，这将可能花费一些时间。它会呈现一个类似于图 2-45 所示的进度条。

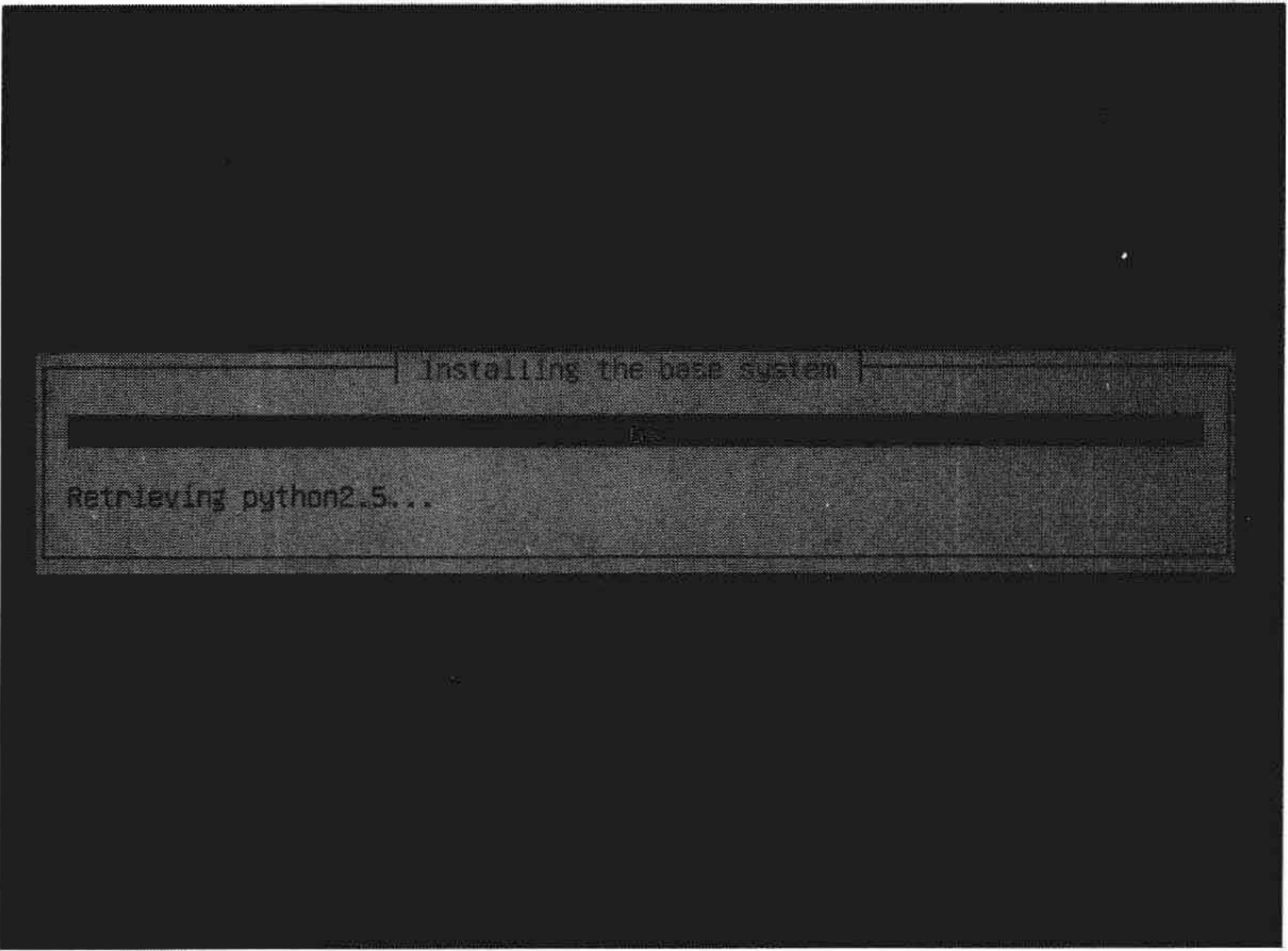


图 2-45 安装基本的系统

接下来，它要求为这个主机创建一个用户。在第 4 章我们会更详细地讨论用户管理，但重要的是要知道 Ubuntu 发行版通过屏蔽 root 密码禁用了 root 用户账号。Root 用户就像 Windows 里的 Administrator，它有访问主机上一切内容的权限。与安装 RHEL 时设置 root 用户的密码不同，在 Ubuntu 上，用户使用一个叫作 sudo 的特别命令享受 root 用户的特权。第 4 章将详细阐述 sudo 命令。

图 2-46 要求输入新用户的全名。

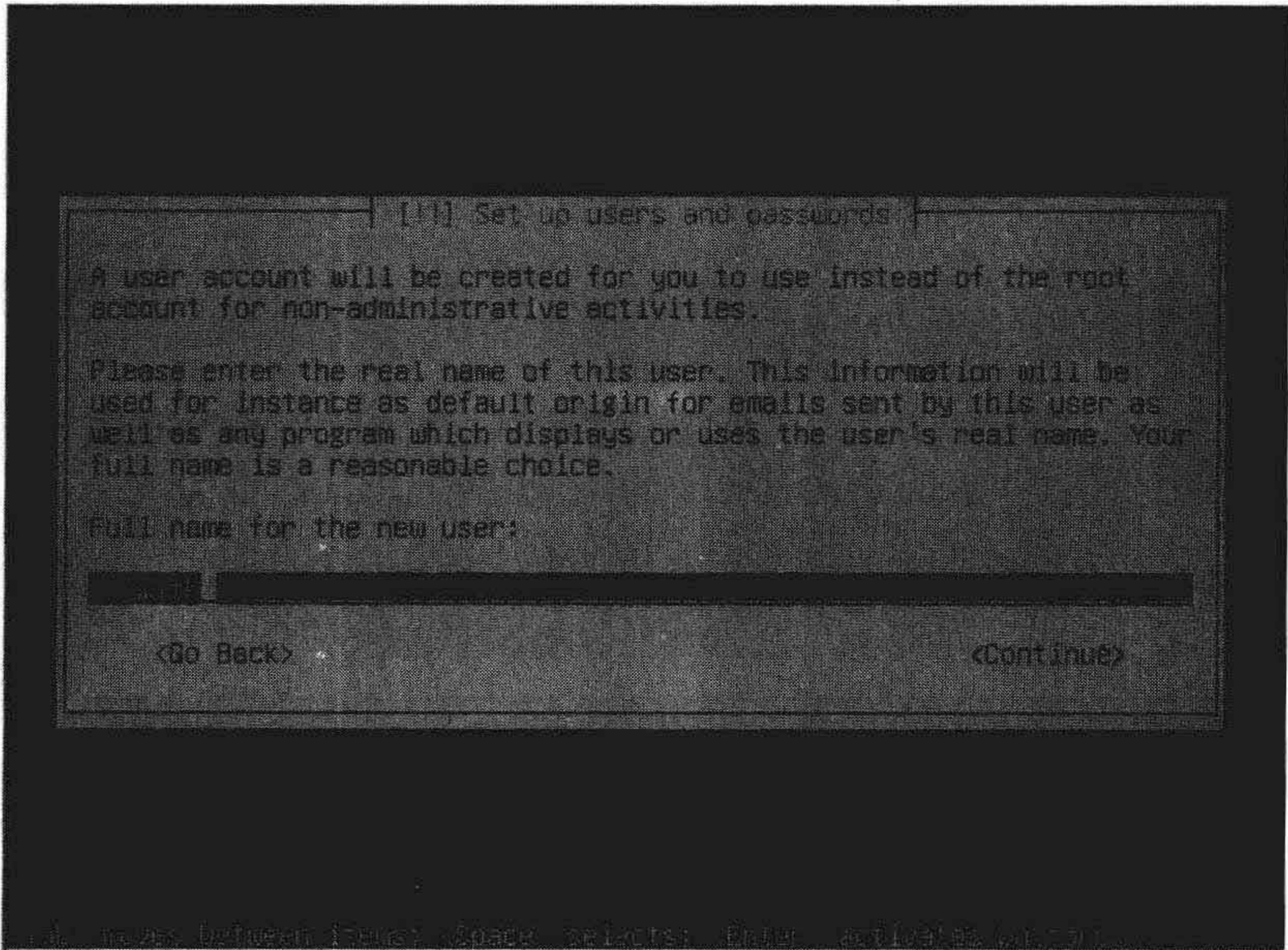


图 2-46 输入新用户的全名

在下一画面，你（见图 2-47）要为新用户设置用户名。

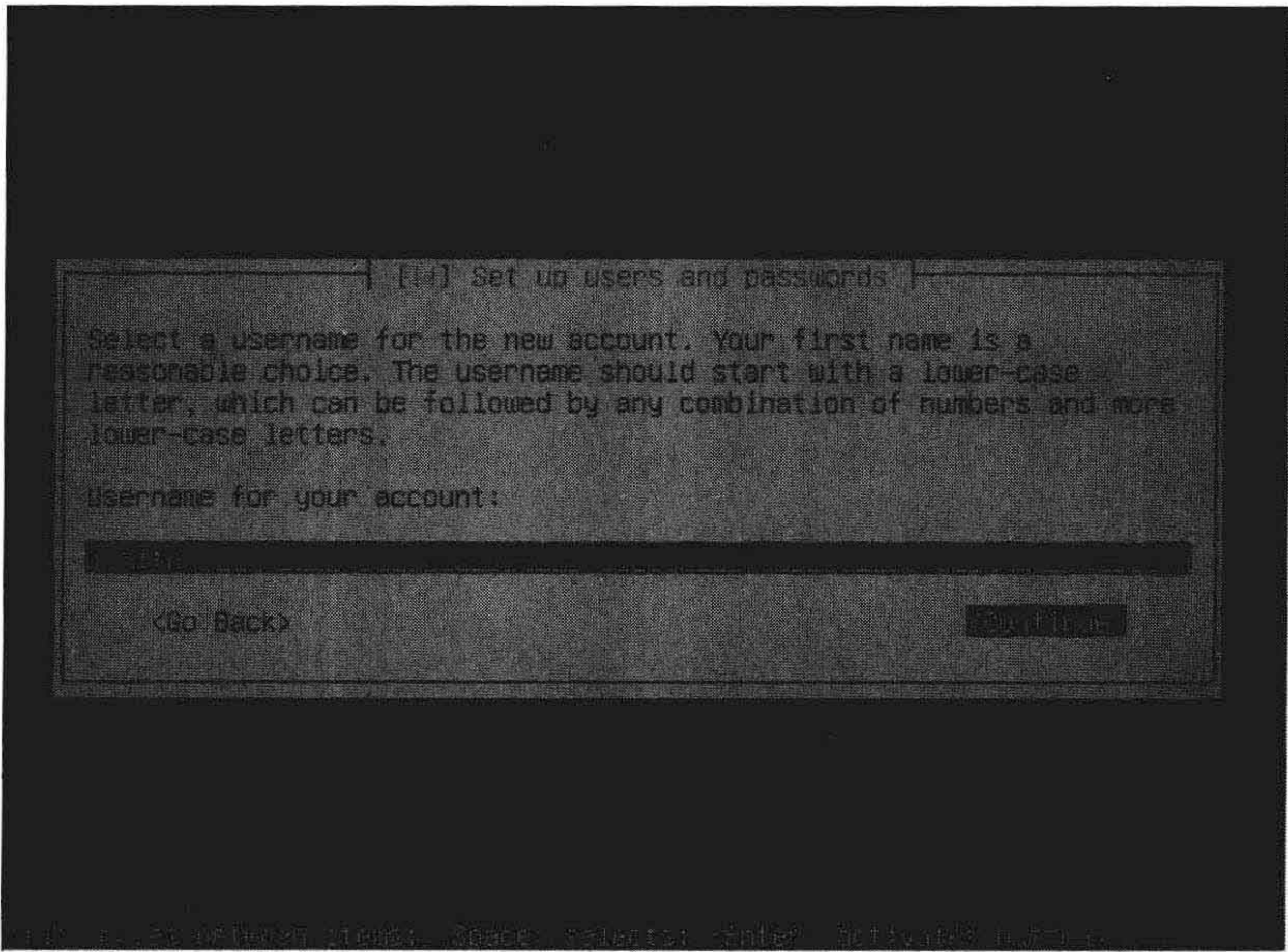


图 2-47 输入新用户的用户名

最后，为新用户设置密码，如图 2-48 所示。同样，正如在 RHEL 安装一节中所讨论的那样，我们推荐设置一个强而复杂的密码。它要求你确认此密码。

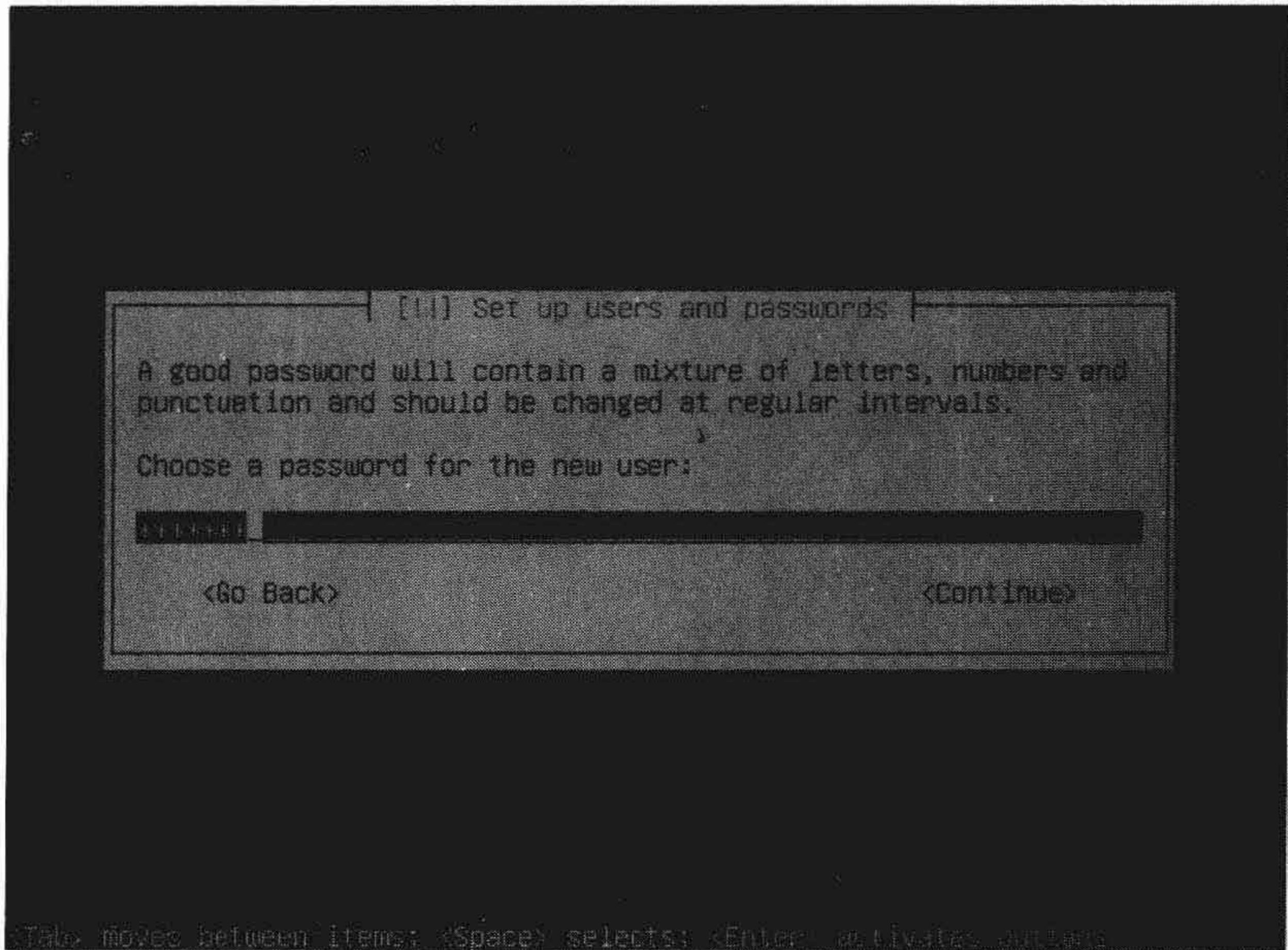


图 2-48 设置用户密码

接下来，它要求填写访问外部世界可能需要的代理服务器的信息。我们暂时忽略它，并继续往下。

然后它问你想要在主机上安装什么应用程序，你要选择程序组。我们选择 DNS、LAMP 服务器（Linux、Apache、MySQL 和 PHP）、邮件（Postfix）和 OpenSSH，如图 2-49 所示。你准备好后，选择“Continue”。

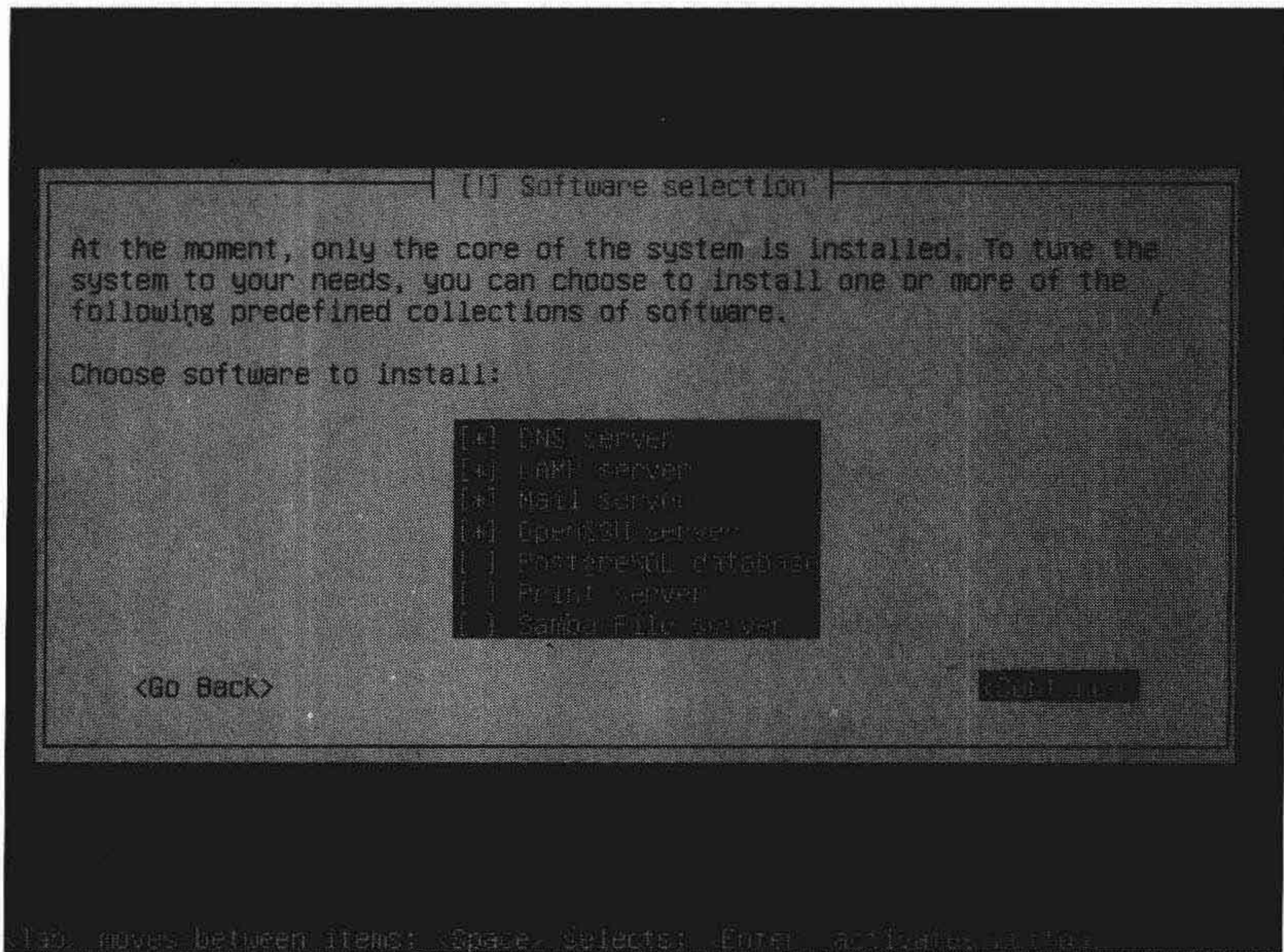


图 2-49 为主机选择应用程序

在本次特定的安装过程中，基于所选择安装的应用程序，它会提出一系列的问题以帮助 Ubuntu 配置或保护所选定的应用程序。每次在 Ubuntu 上安装需要配置输入的新程序时，它都会提示你回答类似的问题。

如图 2-50 所示，它第一次要求你提供 MySQL 数据库的 root 用户的密码。这是 MySQL 安装的主密码，你应输入一个强而复杂的密码。然后它要求重新输入一遍以确认此密码。

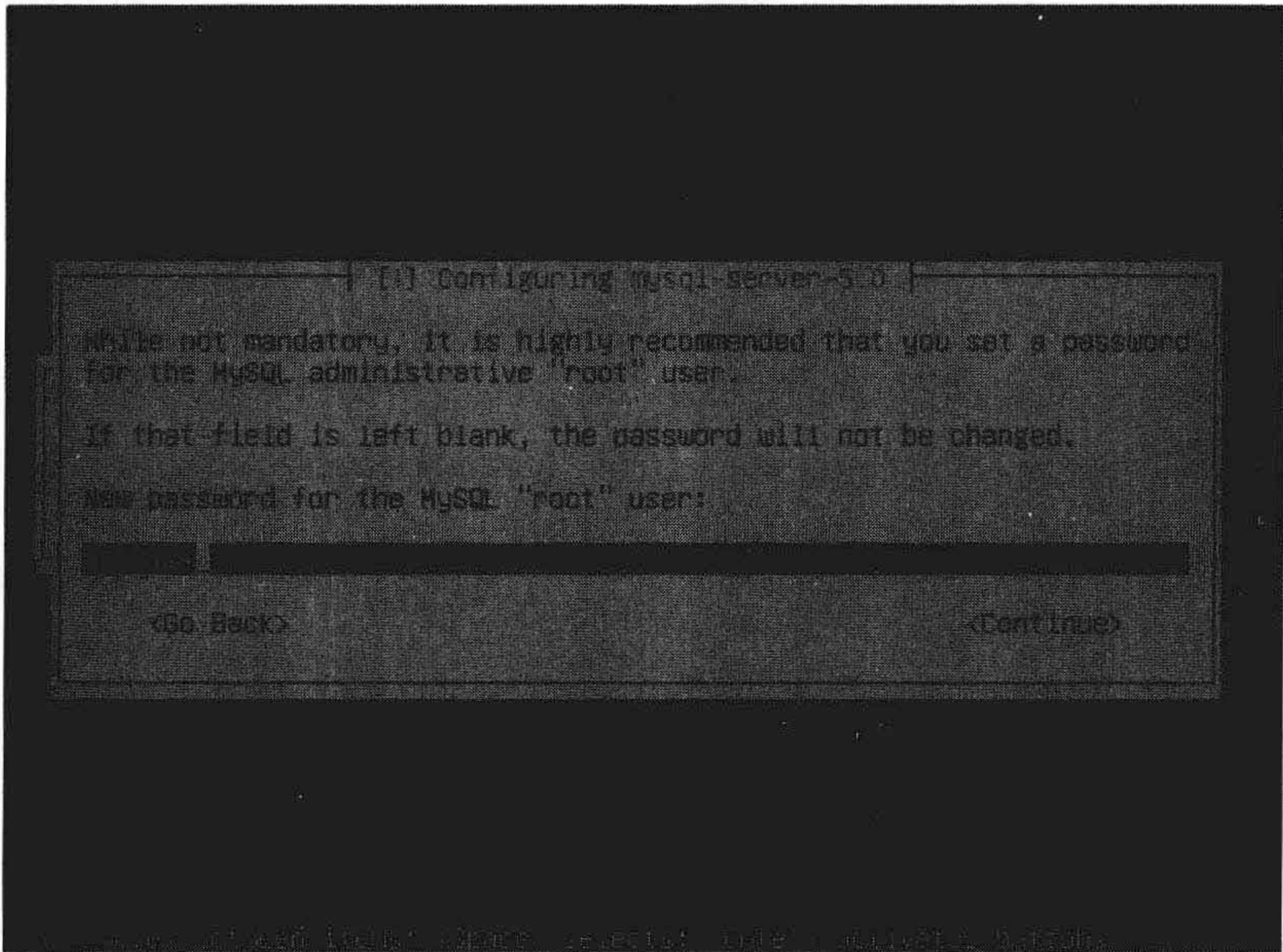


图 2-50 设置 MySQL 的 root 密码

一旦提供了这个密码，它接着就要求你描述邮件服务器的配置。图 2-51 显示了配置选项，而且每个选项都有简短的说明。我们只选择默认选项“Internet Site”。

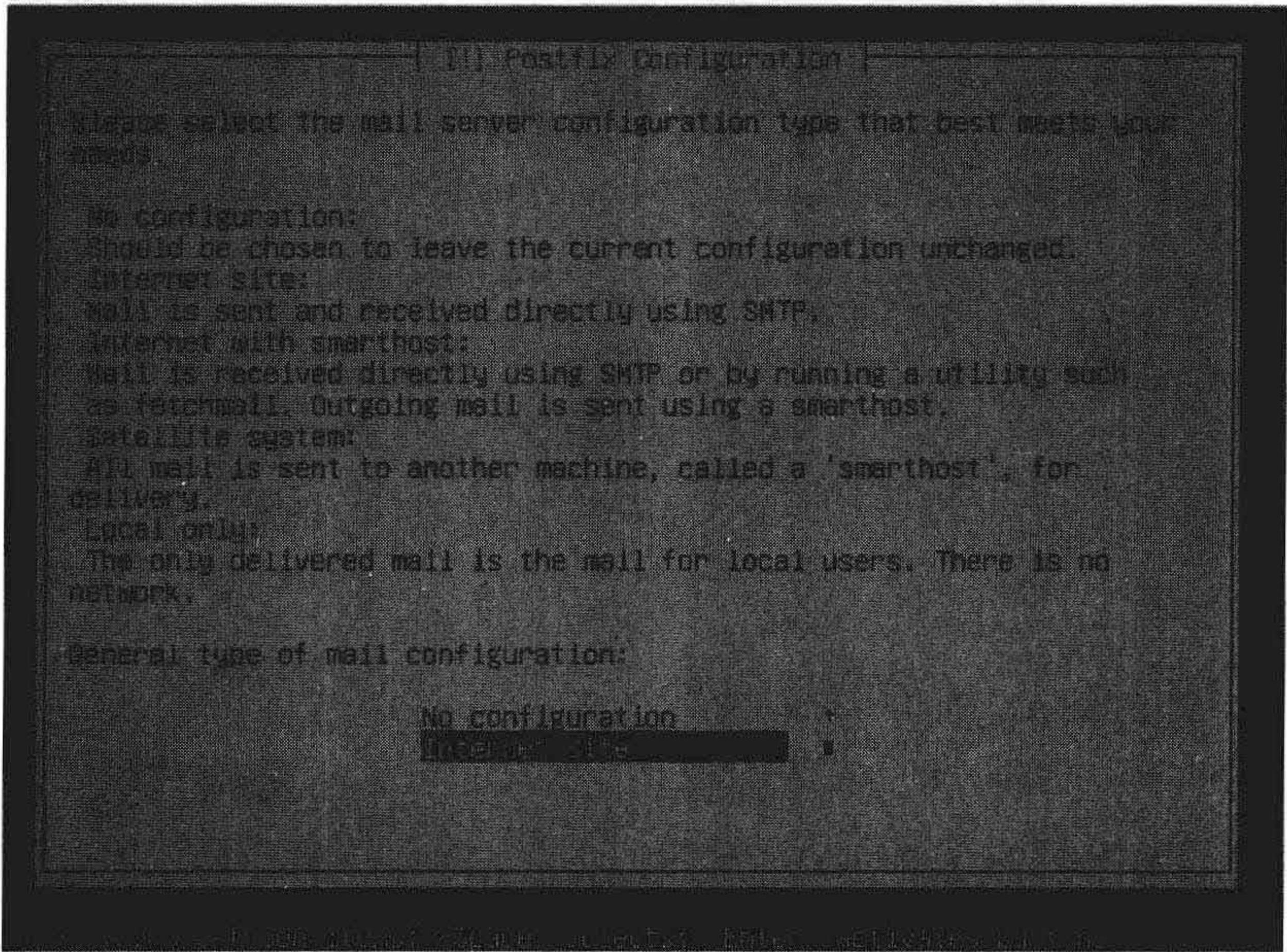


图 2-51 配置邮件服务器

第 10 章会阐述如何配置和保护邮件服务。在这里选择默认选项只能提供一个基本和安全的配置，只能收发域内邮件。

接下来是提供邮件服务器的域名（见图 2-52）。目前你需要输入主机的域名，其他可能的选项我们会在第 10 章阐述。

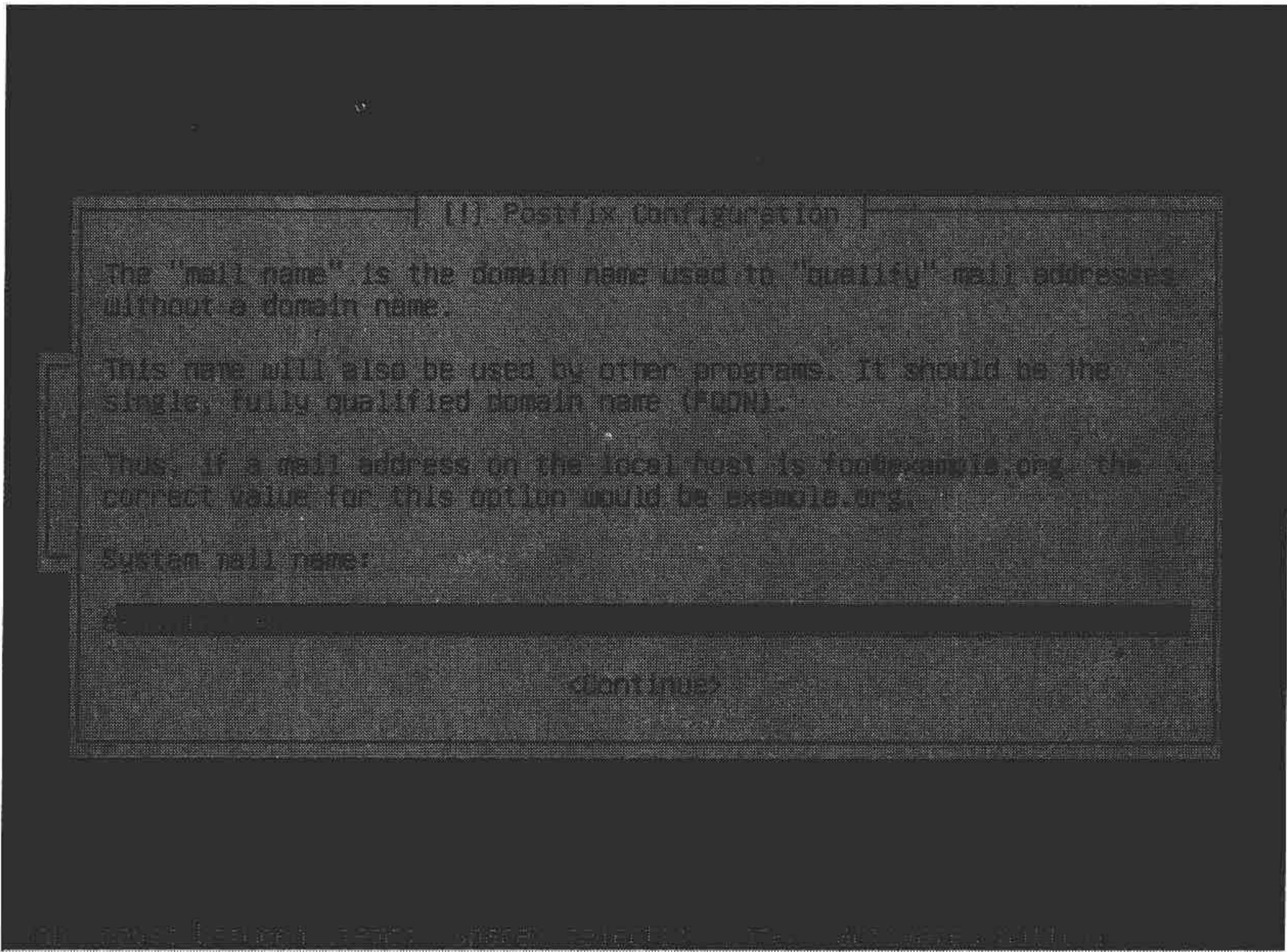


图 2-52 设置邮件服务器域名

如图 2-53 所示，确认计算机时钟被设置为 UTC 之后，安装就接近完成了。主机用 UTC 和时区信息把时间转换为本地时间。

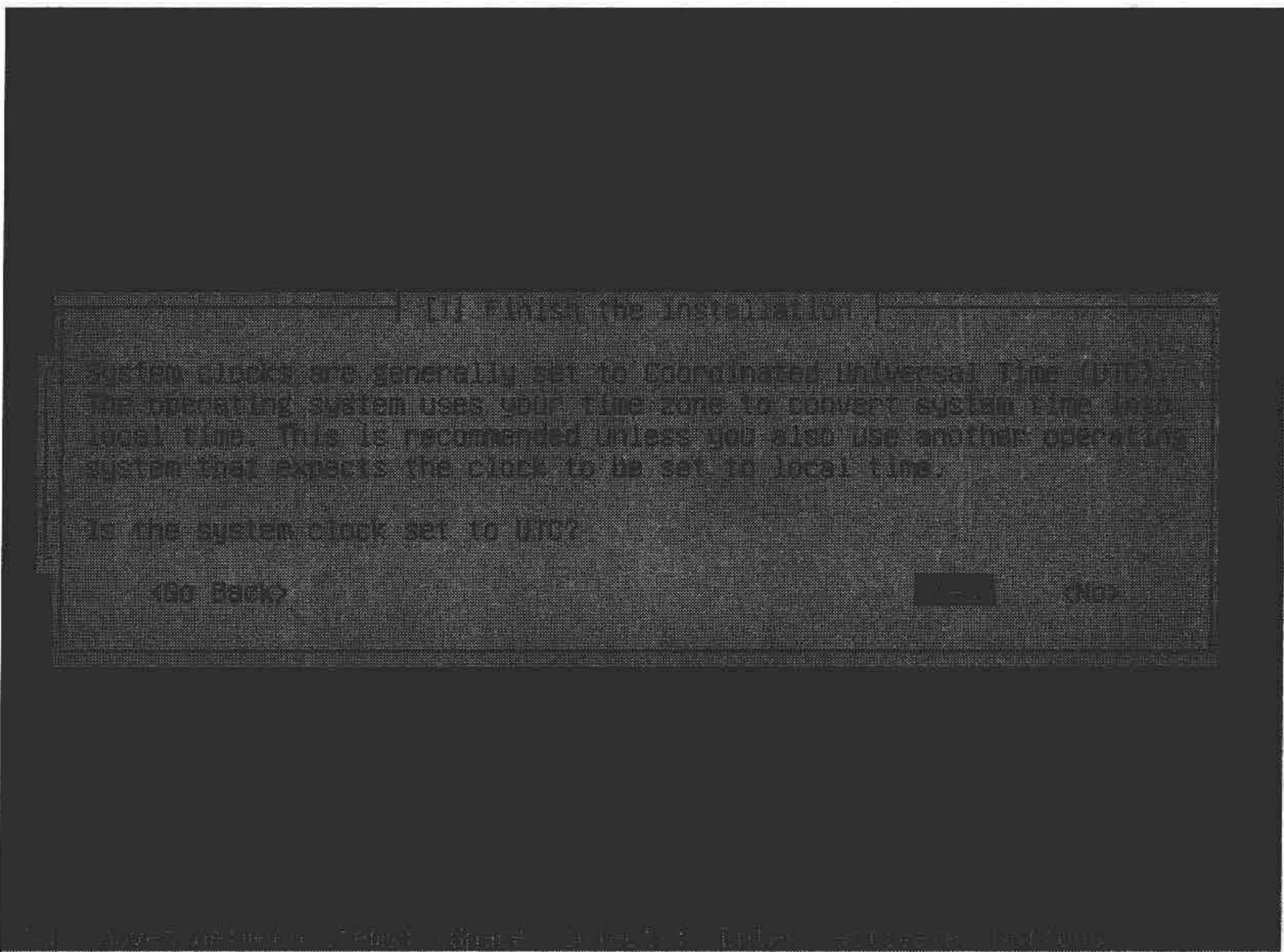


图 2-53 把时钟设置为 UTC

现在安装就完成了，Ubuntu 会通告此事，如图 2-54 所示。

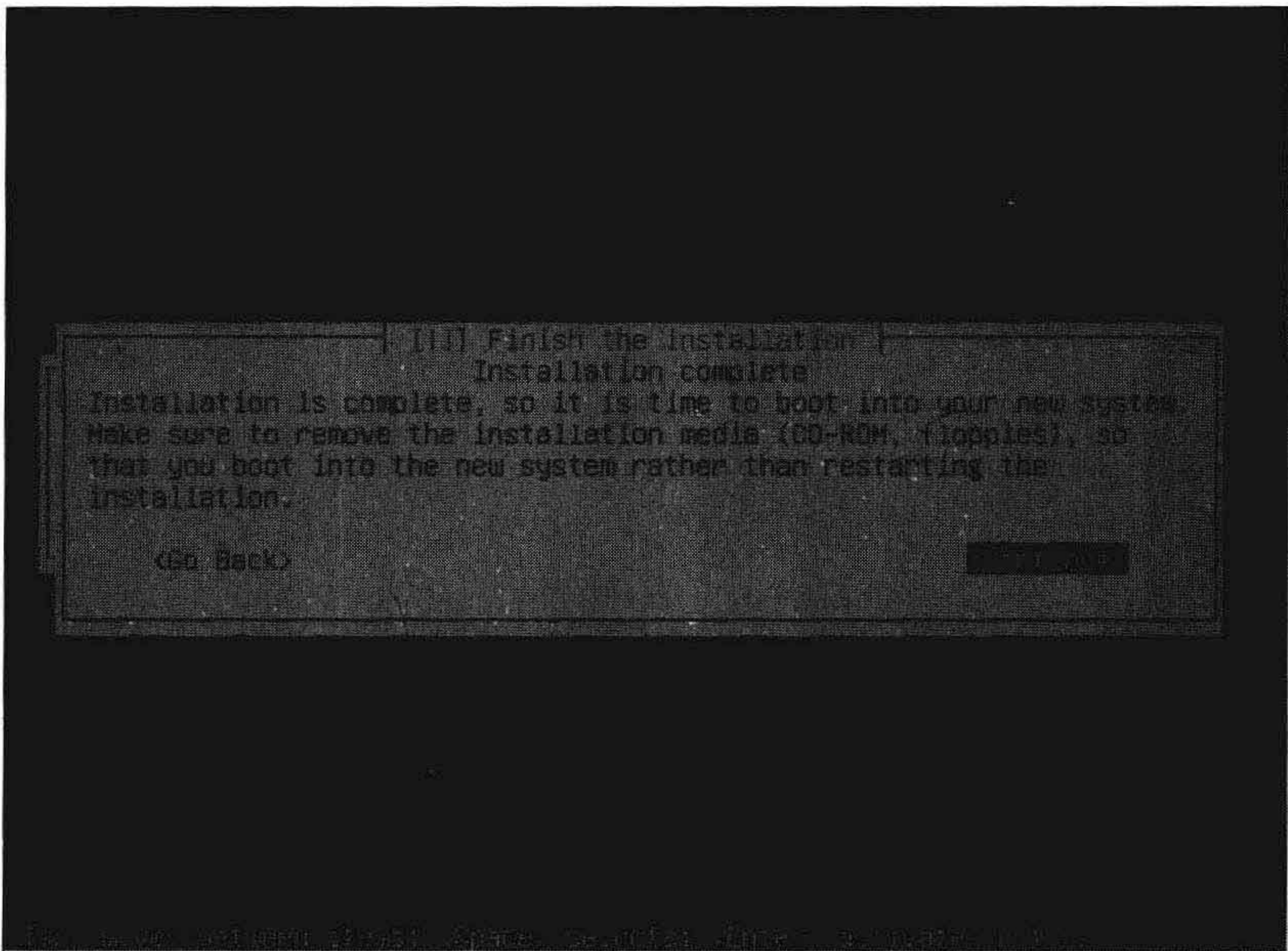


图 2-54 安装完成

CD 或 DVD 将自动弹出，在把它移出驱动器之后，可以选择“Continue”。系统会马上重启，然后你会看到登录提示（见图 2-55）。

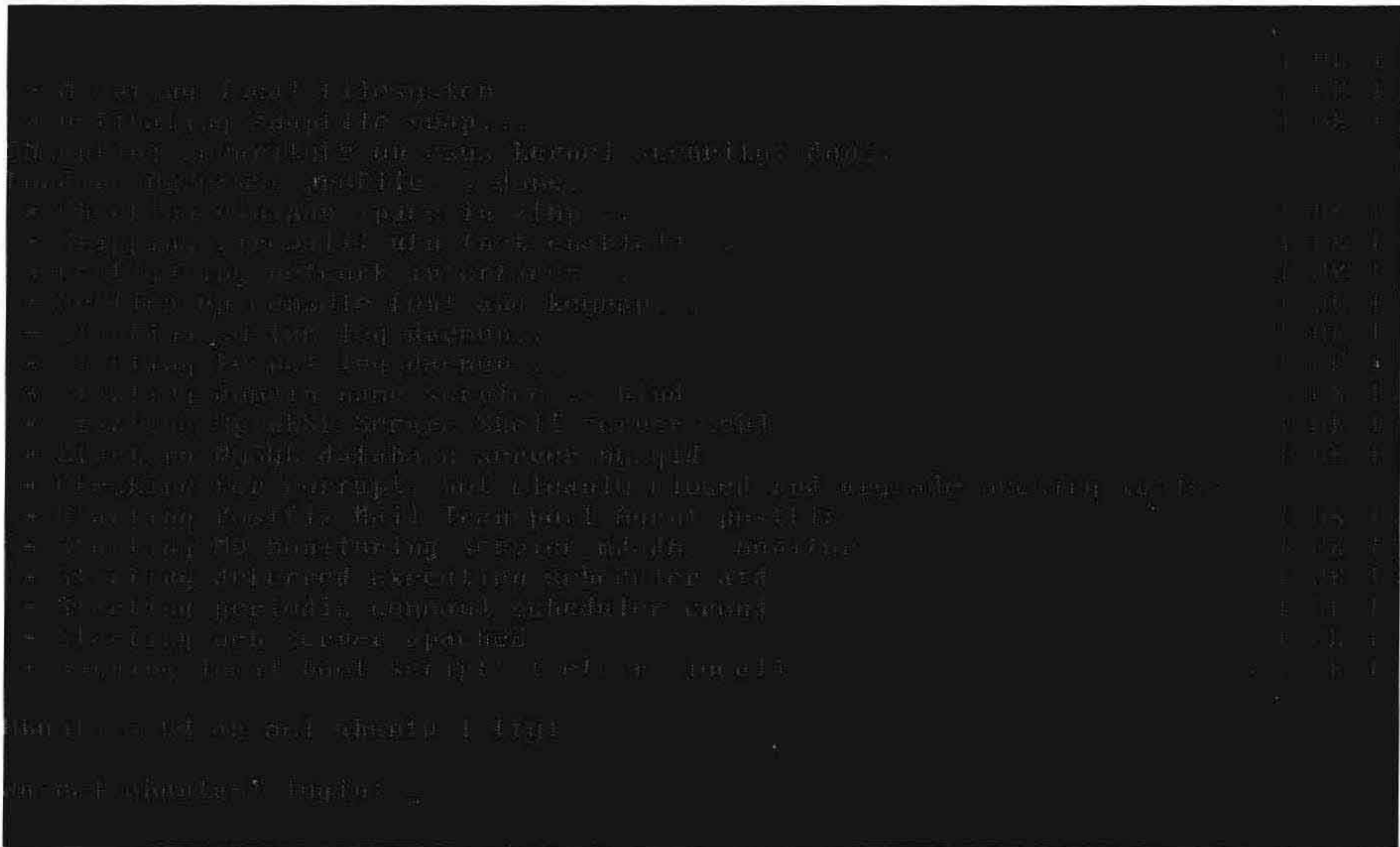


图 2-55 启动到控制台画面

你会注意到 Ubuntu 并没有启动到图形用户界面（GUI），而是到了一个控制台的屏幕。这是因为默认的 Ubuntu 服务器安装过程没有安装 GUI。我们会在第 3 章详细比较命令行和 GUI。

现在你有了一个运行着 Ubuntu，可以使用邮件、DNS 和网络的服务器，它为进一步定制平台做好了准备。

2.4 故障检修

有时安装会因为某种原因失败。最常发生的原因是安装介质问题；很少是由硬件不支持或由故障引起的。

如果安装介质有问题，那么你就可能看到被记录下的读错误，或者安装程序可能会显示一个错误，声明它不能读取某个文件。你应该检查安装 CD 或 DVD 有没有划痕。如果你是从一个 ISO 文件创建的 CD 或 DVD，可能以更低的速度写一个新盘是值得尝试的。介质问题通常在安装过程的同一步骤重复发生。

网络安装还可能因为连接中断而失败，所以你要检查网线是否插好，Internet 连接是否正常。

硬件不支持是较少见的故障类型。例如，如果安装内核不支持磁盘控制器，安装程序将不能访问硬盘。如果发生了这种情况，检查安装盘中的内核版本，核实它事实上支不支持你的硬件。新的发行版本因为支持更多和更新的硬件，也许会好用。

发生在安装过程的随机位置的不可恢复的故障通常预示着硬件的问题，最常见的问题是劣质 RAM 或者是过热。你可以运行一个 RAM 测试器，比如 memtest86 (<http://www.memtest.org/>)，你还应该查证 CPU 和风扇是否正常工作。

2.4.1 诊断信息

如果在安装时需要辅助的诊断信息，你可以访问一个受限的 shell 和安装过程中的一些记录信息。使用这些信息可以更深入地诊断任何可能的问题。

在 RHEL 主机上，ALT+F2 组合键可以访问受限的 shell，ALT+F3 提供安装记录，ALT+F4 提供系统消息，ALT+F5 提供其他消息。ALT+F7 返回到安装过程的 GUI。

在 Ubuntu 主机上，ALT+F2 和 ALT+F3 两个组合键都提供对一个受限 shell 的访问。ALT+F4 提供安装程序的详细安装进度和记录。ALT+F1 组合键切回安装界面。

2.4.2 重新安装

出现问题后，通常我们应该从头开始重启安装过程。因为来自上次安装尝试的文件可能还在磁盘上，所以最好让安装程序重新对分区初始化，一切从头开始。

2.4.3 故障检修资源

遇到故障时，我们要敢于利用大多数 Linux 发行版都存在的社区。可能别人经历过和你

同样的问题，并已经把解决方案归档。这里给出一些可以尝试的资源。

- Red Hat: <https://www.redhat.com/apps/support/>。
- Fedora: <http://forums.fedoraforum.org/forumdisplay.php?f=6>。
- Ubuntu: <http://ubuntuforums.org/forumdisplay.php?f=333>。

2.5 小 结

本章逐步讲述了两个流行的 Linux 发行版的安装全过程。

- Red Hat 企业版 Linux。
- Ubuntu 服务器。

本章还说明了当安装过程中出现问题时你要做什么。下一章将简要介绍使用新 Linux 主机的基础知识。

第3章 Linux 基础



第1章简单地介绍了Linux是什么以及它来自哪里；而在第2章中，我们安装了第一台Linux主机。本章将介绍一些基本的Linux概念和技巧。有些人看到那些带奇怪开关和神秘选项的不可思议的命令，感觉Linux挺吓人的，我们将解码一些必要的命令，并演示这些命令以及它们的功能。

本章关注启动、登入以及在命令行和文件系统中工作与探索，还会介绍一些基本的Linux概念：用户、组、软件包、服务、文件系统以及处理文件和目录的方式。接下来的几章中将会扩展这些概念，介绍那些必须了解的关键活动，以便操作和管理Linux主机。

本章主要讲述以命令行方式运行的指令，介绍命令行的使用方式，并帮助你适应在命令行方式下的操作。这并不是说Linux没有大量图形管理工具可用。如果感觉在类Windows的图形化环境里更舒适，你仍然可以轻易而有效地找到Linux主机的管理机制。对我们将要介绍的大多数命令行工具，都有对应的图形工具。

注：本章是对Linux的一个宽泛的介绍。这不会使你成为专家。更确切一点说，这只是为配置Linux基础架构提供初始的准备。

3.1 准备开始

如果还没有安装Linux，在安装Linux之前尝试Linux命令的最容易的方式就是运行LiveCD。LiveCD是在CD或者DVD上的Linux发行版。为了使用LiveCD，你需要下载一个用户选定的LiveCD的ISO文件类型的镜像，并把它烧录到CD或DVD中。下面的一些网址介绍了如何把ISO文件烧录到CD和DVD。

- <http://pcsupport.about.com/od/toolsofthetrade/ht/burnisofile.htm>
- http://www.petri.co.il/how_to_write_iso_file_to_cd.htm
- <http://help.ubuntu.com/community/BurningIsoHowto>

有了烧录的LiveCD之后，你就可以把光盘插入计算机并重启。大多数计算机会检测到LiveCD，并提供从CD或DVD启动的选项。

■注：如果主机没有提供从 CD/DVD 启动的选项，就可能要调整 BIOS 设置来修改启动顺序，以使 CD 或 DVD 在硬盘驱动器之前启动。

LiveCD 会载入一个处于工作状态的 Linux 发行版供用户试用。一般情况下，这不会往主机安装任何东西，原来的配置在移除 CD/DVD 并重启之后仍然可用。

多种发行版都有 LiveCD。一些用起来不错的发行版 LiveCD 包括以下几种。

- Ubuntu: 可以在 <http://releases.ubuntu.com/> 找到 Ubuntu 的 LiveCD，称为 Desktop CD。
- Fedora: 可以在 <http://fedoraproject.org/en/get-fedora-all> 找到 Fedora 的 LiveCD（称为 Fedora Desktop Live Media）。
- CentOS: 对最新的 CentOS 5 发行版可用的 LiveCD 来自 <http://isoredirect.centos.org/5/isos/i386/> 上列出的镜像网址。

■注：在 http://en.wikipedia.org/wiki/Comparison_of_Linux_LiveDistros 上有一个对某些 Linux LiveCD 的比照表。

登入

Linux 主机或者 LiveCD 启动之后会呈现一个命令行或图形用户界面形式的登录提示界面。

如图 3-1 所示，这是 Ubuntu 主机的经典的命令行登录提示界面。而图 3-2 则是 Red Hat Enterprise Linux 主机的图形登录界面。

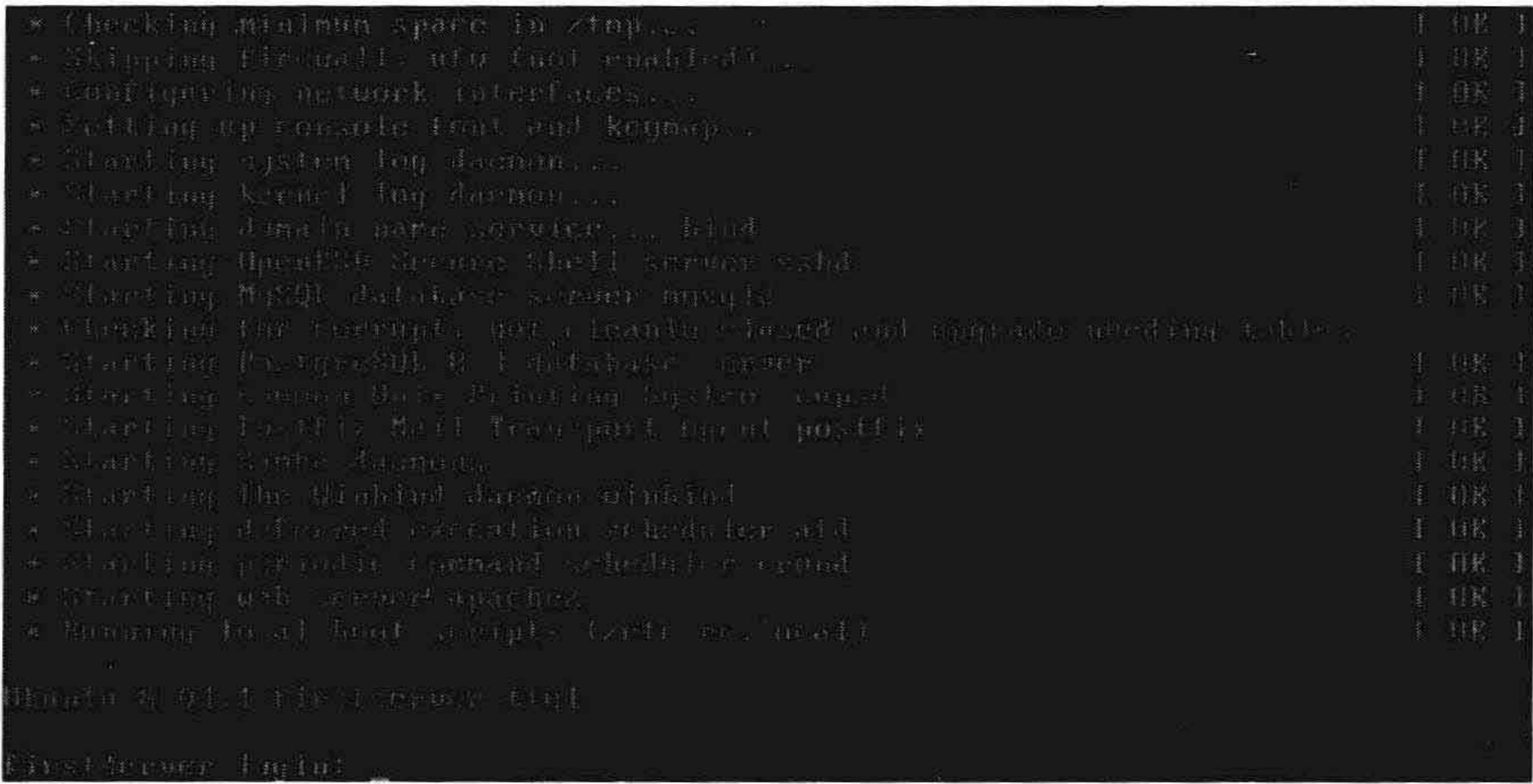


图 3-1 命令行登录提示

■注：如果最初的画面和这些稍有不同，请不要恐慌，因为在不同版本之间会有一些细微差别。



图 3-2 图形登录提示

在登录提示中，需要提供用户名和密码，或者是类似的验证形式。（与 Windows 一样，Linux 也可以使用智能卡、令牌或者其他机制来验证用户。）

如果刚安装好 Linux 主机，安装过程中应该提示过创建一个用户，现在就可以使用那个用户登入。如果正在用 LiveCD 试用 Linux，那么可以看到它提供了一个默认的用户名和密码，或者甚至可以自动登入。例如，Ubuntu 的 LiveCD 有一个默认的用户名 ubuntu 和密码 ubuntu，通常会自动登入。如果没有看到默认的用户名和密码，那么可能需要查看 LiveCD 的在线文档，或者它可能提示创建一个用户名和密码。

在主机验证权限之后，你就登入成功了。根据用户的配置，主机或者显示一个命令行，或者显示一个 GUI 桌面环境。

3.2 Linux 与 Microsoft Windows

本节的标题听起来可能有点我们要给出一个专业的高下对比的意味。不过，更多的是 Linux 和 Microsoft Windows 的相似之处，而不是它们的区别。Windows 和 Linux 二者都是操作系统，尽管在许多技术方面不同，但它们有很多相同的概念。它们也许对某些概念用了不同的名字；例如，Windows 称用户的个人信息、配置和磁盘空间为 profile（配置档案），而 Linux 领域有一个相似的概念为 home directory（主目录）。概括来说，不论如何，它们总体的概念是相同的。因此，我们要分析这些相似点，这有助于利用已有的 Windows 知识来理解 Linux 的相关概念。

本书的目的是研究如何与 Linux 主机交互。基本的界面有两种：GUI 桌面和命令行。本书将探究这两种界面。

3.2.1 GUI 桌面

Linux 和 Windows 二者都有图形化用户界面。Windows（如果没有问题的话）启动进入

GUI 界面，在该界面可以打开命令行的窗口。与 Windows 不一样，Linux 既可以启动进入 GUI 界面，也可以启动进入命令行界面。启动之后，还可以在这两种模式之间转换，至于如何做到这一点，我们会在稍后的“命令行”一节讨论，更详细的介绍会在第 5 章进行。

Linux 上的 GUI 界面是几个程序的组合。基本的程序叫 X 桌面系统（X Window System）（你还会看到它被称为 X11 或直接称 X）。X 程序具有提供潜在“窗口”式环境的能力。

■注：不用对 X 的安装或管理担心。在安装 GUI 桌面时，你的发行版通常会安装它。如果没有安装 GUI 桌面，比如只安装一个服务器，X 就不会安装，这样的话，一般就通过命令行界面和 Linux 交互。在第 2 章安装的 Ubuntu 服务器就是一个没有默认加载 GUI 的发行版的例子。

然后在 X 的上面添加一个桌面环境，以提供外观支持和桌面功能，比如工具条、图标和按钮等诸如此类的东西。Linux 上流行的桌面环境主要有两种：Gnome 和 KDE。大多数发行版都把其中之一作为缺省选项；例如，Debian、Ubuntu、Red Hat 和 Fedora 发行版默认为 Gnome，而 Ubuntu 衍生的 Kubuntu 和 SuSE 都默认为 KDE。

■提示：根据 Linux 一贯的灵活性原则，在所有这些发行版上都可以把默认桌面环境改为 KDE。

图 3-3 显示的是 Ubuntu 发行版上的默认 Gnome 桌面。



图 3-3 Hardy Heron 版本的 Ubuntu Gnome 桌面

3.2.2 命令行

在 Linux 领域里，命令行是可以利用的最强大工具之一。本书大部分焦点都在命令行上。至少某些管理任务要在命令行上进行，因此，能够理解和使用命令行是很重要的。的确，在

某些情况下没有 GUI 环境可用。如果 GUI 环境不能正常工作了，你就要能使用命令行管理主机。命令行还提供一些强大的工具，它们使管理任务变得更快、更高效。

■注：这不是说要忽略 GUI。我们还将说明如何使用 GUI 工具管理 Linux 主机。

让我们看一看 Linux 命令行。访问命令行的方式有好几种。如果主机已经启动到命令行提示符，如图 3-1 所示，那么就可以直接登入并使用它。

在 Gnome 或 KDE 图形界面上有两个选项。第一个是用虚拟控制台——一种大多数 Linux 发行版上都默认运行的管理控制台。或者可以运行像 Gnome Terminal 或 Konsole 的终端模拟程序。

■注：终端模拟器（terminal emulator）是一种在另一个应用程序里面模拟文本终端的工具。例如，当你在 Windows 中启动一个命令行或者命令行 shell 时，就已经启动了一个 Windows 终端模拟器。

为了从 Gnome 或者 KDE 图形界面里运行虚拟控制台，请使用组合键 Ctrl+Alt 和 F1 到 F6 其中的一个键。可以打开的每一个窗口都是一个新的虚拟控制台。6 个虚拟控制台都可用，你可以使用 Alt+F1 到 F6 循环浏览这些控制台，或者使用 Alt+左箭头键或 Alt+右箭头键来回地浏览这些窗口。这些终端不分主次且互相独立。要回到 GUI，请使用 Alt+F7。

■提示：如果没有运行 GUI 界面，那么虚拟控制台仍可用。可以使用 Alt+F1 到 F6 键和 Alt+左箭头键和 Alt+右箭头键操纵它们。

还可以运行终端模拟器。例如在 Gnome 里，单击“应用程序”菜单，打开“附件”标签并选择“终端”程序。这会启动 Gnome 的“终端”程序，如图 3-4 所示。



图 3-4 运行 Gnome 中的“终端”程序

在 KDE 里稍微有点不同。在 KDE 的早期版本里，运行 Konsole 程序的方式是：单击“应用程序”，打开“系统工具”，然后选择 Konsole 程序。对于版本 4 以上 KDE，方式变为：单击“应用程序”，选择“系统”，然后再选择 Konsole 程序。

1. Shell

呈现的是什么命令行取决于用户运行的是什么 shell。shell 是与操作系统以及主机内核的接口。例如，Windows XP 上的命令行也是一个 shell。每个 shell 包含一组内置的命令，使你可以与主机交互（这些是发行版安装的附加工具所增补的程序）。

可用的 shell 有许多种，最常见的是 Bash（或 Bourne-again）shell。许多发行版默认使用它，包括流行的 Red Hat、Ubuntu 和 Debian 发行版。

■注：第 1 章和第 2 章对具体的发行版进行了详细讨论。

本章将使用 Bash shell 演示所有的例子，因为它是最容易找到的 shell。

2. 命令行提示符

登入 Linux 主机之后，你应该看到一个类似于下面这样的提示符。

```
jsmith@au-mel-rhel-1 ~$
```

那么这是什么意思呢？好，我们分解一下。

```
user@host directory$
```

在大多数 Linux 发行版里，基本的提示符由所登录的用户名、主机名、当前目录和\$符号构成，这表明你是以何种用户身份登录的。

■提示：你可以定制提示符使它包含其他信息，添加或修改颜色，或者实现各种其他的选项。更多信息请参考<http://tldp.org/HOWTO/Bash-Prompt-HOWTO/>。

在我们的例子中，jsmith 是登录的名字，接下来是@符号，然后是登入的主机名，即 jsmith 在 au-mel-rhel-1 主机上。

■注：这种提示符看起来像 e-mail 地址，这是有原因的。这就是 e-mail 开始的方式——在互连的 UNIX 主机上有登录账号的人们互相发消息。@符号因此在 1971 年被首次使用。可以到<http://openmap.bbn.com/~tomlinso/ray/firstemailframe.html>上了解相关内容。

接着看到的是一个~符号，这是对主目录的省略写法。在 Linux 主机上，大多数用户都有一个特别的目录，叫主目录（home directory），它是创建用户时被创建的。像 Microsoft Windows 的用户配置档案（profile）一样，用户的偏好和配置文件及数据都存储在这个目录里。任何时候只要看到使用了~符号，就表明是主目录的快捷方式。之前曾提到过主目录，它粗略等同于 Documents and Settings profile 与 My Documents 文件夹这两个 Windows 概念的组合。通常在/home 目录下可以找到主目录。

■注：Linux 是多用户操作系统，多个用户可多次登录，可同时工作。与 Windows 一样，用户可以有它们自己的环境、存储、访问控制和权限。

最后看到的是\$符号。它表明用户的类型；所有普通用户默认以\$作为提示符。有一个特殊用户 root，它的提示符使用#符号。

```
root@au-mel-rhel-1 ~#
```

root 用户就是超级用户。在 Windows 中，我们称这种用户为 Administrator。与 Windows 中的 Administrator 用户一样，root 用户可以控制和配置一切。所以如果看到#符号，你就知道自己正以 root 用户身份登录。

在有些发行版里，你可以以 root 用户的身份登入，通常在安装过程中它会提示你为 root 用户指定一个密码。其他大部分发行版，特别是 Ubuntu，禁用了 root 用户的密码。在 Ubuntu 上假定绝不会使用 root 用户，而会使用一个叫做 sudo 的特别命令。sudo 命令使你在运行命令时有 root 用户的特权而不用以 root 用户登入。我们会在第 4 章介绍 sudo 命令。为了使用 sudo 命令，你只需直接键入 sudo 和希望运行的命令。通常这时会提示你输入密码，如果输入的密码正确，命令就会被执行。

```
$ sudo passwd root
```

这条命令将修改 root 用户的密码，这正是 Ubuntu 上解禁 root 用户的一种方法。

root 用户是全能的，可以做任何事情。因此，当以 root 用户登录时，容易发生意外的错误而删除数据或者破坏应用程序和服务。所以，为了保险起见，绝不要以 root 用户登录。我们在稍后第 4 章会介绍不使用 root 用户登录而管理主机的方法。

■注：近年来我们引入了其他安全控制措施以减少对 root 用户的依赖，并提供更多粒度的安全控制。这些控制包括像 SELinux 和 AppArmor 之类的工具，这些工具曾在第 2 章简要介绍过。

3. 键入第一条命令

你现在可以尝试输入命令了。一条命令可能是一个可执行的二进制代码（就像 Windows 可执行文件或 EXE 文件），或者是作为 shell 的一部分提供给我们。下面键入一条叫做 whoami 的命令，并按回车键来执行它。

```
$ whoami
jsmith
```

whoami 这条命令返回所登录的用户名。你可以看到主机返回了 jsmith。这表明我们以用户 jsmith 的身份登入了这台主机。

每个 shell 都包含一系列的内置命令和操作，以便帮助用户使用命令行。现在来尝试其中的一个。仍以 whoami 命令开始，不过这次包含一个拼写错误，键入错误的命令名。

```
$ whoamii
```

■注：本书将把 shell 的提示符缩写为只有最后的提示符符号\$或者#。

接着按回车键运行它，我们发现 Bash 返回下面的回应。

```
-bash: whoamii: command not found
```

那么发生了什么？哦，Bash 说不存在叫做 `whoamii` 的命令。这个问题可以解决。让我们开始纠正这个命令。使用上箭头键恢复上次键入的命令，你将会看到上一次的命令回到了命令行。

```
$ whoamii
```

Bash 有个好用的命令历史功能，它保存许多以前键入的命令。Bash 允许通过使用上下箭头键定位这些命令。

提示：保存的历史命令总数是由用户配置的，你可以通过 `history` 命令来操作。现在输入 `history` 命令来查看一下命令历史。如果刚开始使用，你可能发现历史是空的。在这种情况下，请你使用几条命令后再试。你会看到一个有编号的行列表，显示先前键入的命令。你可以通过输入挨着命令的编号以及前缀 `!` 符号重新找回任何一个命令，`!12` 将重新找回并执行历史中编号为 12 的命令。

你还可以使用左右箭头键顺着命令行移动光标编辑命令。使用右箭头键移动到命令结尾并删除多余的 `i`，余下的命令如下所示。

```
$ whoami
```

现在按回车键，就会在命令行上看到结果。

```
jsmith
```

这次正确的命令 `whoami` 又一次返回所登入的用户名。

提示：另一个有用的 Bash 功能是自动补全。开始键入一条命令，然后按 `Tab` 键，Bash 会搜索你的路径，试图找到你所希望发布的命令。键入的字符越多，`Tab` 键搜索的范围越窄。

补充知识：路径

如果 Linux 告诉你不能找到一个二进制代码或者一个命令，可能是命令名中包含拼写错误，或者它不能找到那条特定的命令。与 Windows 一样，Linux 在执行命令时搜索一些路径，试图找到那条命令。大多数发行版默认设置一个缺省路径，它通常包括有代表性的位置，这些位置包含了可执行的二进制代码。多数情况下都不需要设置路径，采用默认路径就行了。如果想改变路径，需更新一个称为 `$PATH` 的环境变量。我们会在第 5 章介绍环境变量。

3.3 远程访问

前两节介绍了 GUI 桌面和命令行。在这两种情况下，我们都假定用户是在本地登入主机（即坐在屏幕和键盘前直接往主机输入命令）。但是大多数情况下人们都是远程访问 Linux 主机。对那些作为服务器的 Linux 的主机更是如此，它们可能寄存在数据中心或者另外一个地理位置，或者存放在一个箱子或机柜里。很多情况下，这些主机甚至没有附带显示器和键盘，只能通过网络访问。

使用 Linux 很容易远程连接到这些主机来管理和操作它们。你可以使用很多不同的方法进行远程连接。这些方法使用了一种桌面共享协议，如虚拟网络计算（Virtual Network Computing，通常称为 VNC）协议、通常为 Windows 主机提供远程访问的远程桌面协议（Remote Desktop Protocol，RDP）以及被广泛使用的安全 shell 协议（Secure Shell，SSH）。

使用 SSH

很快就要考虑使用 SSH 提供对 Linux 主机的远程命令行访问。使用 SSH 还可以访问 GUI 桌面，但这些将在第 9 章讨论。

SSH 既是应用程序也是安全协议，它有多种用途，但主要用于主机的远程管理。Linux 主机所提供的 SSH 是一个称为 OpenSSH（参见 <http://www.openssh.com>）应用程序的开源版本。

SSH 以客户端-服务器的模式连接 TCP/IP 网络。正在连接的主机是客户端。例如，如果通过膝上型电脑连接一个远程主机，这台膝上型电脑就是客户端。要连接到的主机称作服务器，它接受并管理这条连接。

使用 SSH 的远程连接是加密的，需要一个密码或者公钥密码机制的身份验证。为了建立一个 SSH 连接，你需要知道 IP 地址或者远程主机的主机名，然后在客户端发起一个连接，它通过 TCP 连接连接到端口 22（这个端口可以修改，我们会在第 9 章说明如何修改）。

■注：可能你以前见过 IP 地址和主机名，但还没遇到过端口。端口是服务（如 SSH）所使用的通信端点。端口号可从 0 到 65535，比如大家都知道的 80 是 HTTP，25 是 SMTP，21 是 FTP。1 到 1023 号端口一般保留给系统服务，而 1024 号及以上的端口（或称 ephemeral ports）的分配更随意些。我们会在第 6 章深入更多相关的细节。

连接开始后，服务器会提示客户端提供用户名和身份凭证，如密码。如果输入的用户存在，提供的凭证正确，客户端就被允许连接到此服务器。

大多数发行版都默认安装 Secure Shell，并默认启动服务器程序。Secure Shell 服务器或 Secure Shell 后台程序（在 Linux 的领域里 server 也叫后台程序，daemon）允许远程连接到主机的命令行或 GUI。

你可以通过命令行或者某个客户端使用 SSH。通过命令行使用叫 ssh 的命令可以建立客户端的连接。大多数 Linux 和类 Unix 操作系统（例如 Mac OS X）都安装了 SSH，并提供 ssh 命令。使用 ssh 命令要指定用户名和想要连接的主机，并用 @ 符号分隔它们，如列表 3-1 所示。

列表 3-1 SSH 连接

```
$ ssh jsmith@us-ny-server-1.example.com
Password:
```

在列表 3-1 中，我们连接到一个叫 us-ny-server-1.example.com 的主机，使用的用户名为 jsmith，然后系统提示输入密码。如果输入的密码正确，我们就会登入远程主机的命令行界面。

■注：实际上，如果你原样运行此命令将不会有结果，因为主机 us-ny-server-1.example.com 不存在。如果想测试的话，你就需要指定一个实际存在的主机。

还有各种各样的客户端和终端模拟器可以利用，例如流行的免费 PuTTY 客户端

(<http://www.chiark.greenend.org.uk/~sgtatham/putty>), 它运行在 Windows 上 (Linux 上也有)。
SSH 客户端可以在 GUI 界面里把文本终端用作 Unix 或 Linux 主机的命令行。图 3-5 显示了 PuTTY 客户端的配置画面。

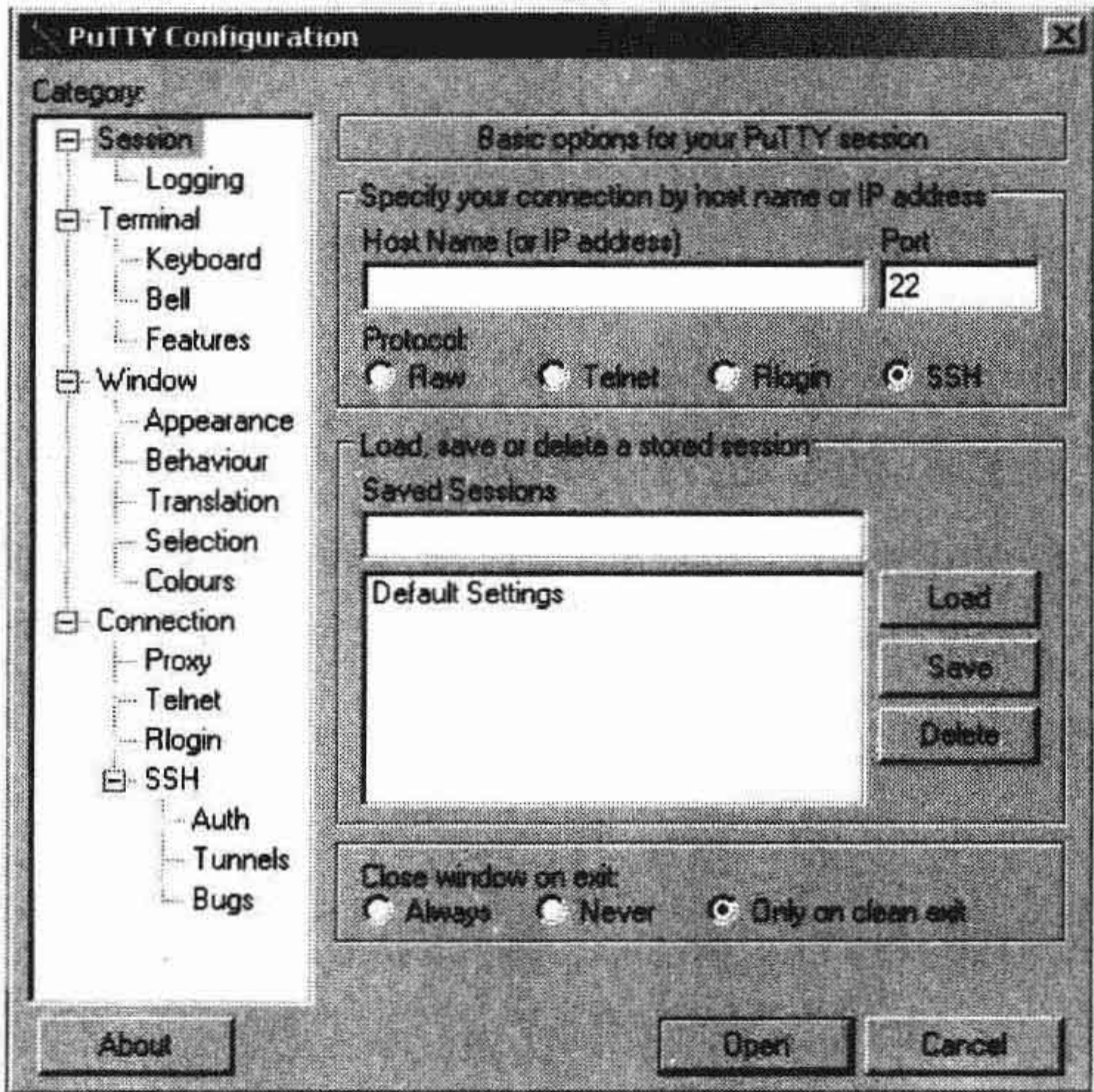


图 3-5 PuTTY 客户端

像 PuTTY 这样的 GUI 客户端的使用方式很简单。对于命令行, 需要指定主机名 (或者 IP 地址) 和要连接的主机的端口。对于类似 PuTTY 的客户端, 你还可以做些有用的事, 比如保存连接, 这样就不需重新输入主机名了。

如果想保持桌面环境运行 Windows, 我们推荐下载像 PuTTY 这样的客户端。这样你就可以在一个舒适的环境里远程连接和管理 Linux 主机。基于 Unix 的 Mac OS 内置了一个 SSH 客户端。

提示: 对于像 Windows Mobile、Symbian 和 Apple iPhone 这样的操作系统, 也存在 SSH 客户端。经常旅行的人在忙碌的时候可以使用它连接到它们的 Linux 主机。

3.4 获得帮助

那么如何在 Linux 主机上获得帮助呢? 你可能想: “不会是使用 F1 键吧?” 呃, 实际上可以。在 Gnome 和 KDE 的图形界面里, F1 键会调出界面的帮助文本。但在命令行里, 同样有很多各种各样的工具, 它们的设计目的就是说明操作方式、帮助查找命令以及解释命令里的可用选项。

最简单的方式是查看命令或程序的 man 页 (manual page 的简写)。man 页告诉你该命令可以做什么、有什么选项以及其他各种相关信息。通过键入 man 和想要查看的命令名可访问 man 页, 如列表 3-2 所示。

列表 3-2 man 命令

```
$ man ls
```

这条 man 命令会返回一个文档, 描述 ls (或者 list) 命令和它的各种选项。

■注：ls 或 list 命令列出主机上的文件和目录。本章稍后将使用 ls 命令展示更多有关文件的内容，所以请稍加等待。

如果对一条命令有疑惑，它的 man 页就是寻找帮助的第一个地方。不是所有的命令都有 man 页，如果某条命令不存在 man 页，就会得到一个错误信息。这种情况下，尝试在命令上追加--help 开关通常会有用，如列表 3-3 所示。

■注：开关是可被追加到某些命令上的命令行选项。开关通过一个短划线或两个短划线和开关的单字母缩写或开关名指定，例如-l 或--name。

列表 3-3 --help 开关

```
$ ls --help
```

■提示：通过 man 命令还可以得到一个不错的对 Linux 的概要介绍。要看这个介绍，请使用命令 man intro。

使用-K 选项还可以搜索所有包含某个关键词的 man 页。

```
$ man -K user
```

这将搜索所有 man 页中的关键词 user，最后返回包含该关键词的所有 man 页。然后系统提示你查看每页，略过某页到下一页，或者退出搜索。

这样的搜索可能有点慢，因为主机中通常有很多 man 页，所以有两种更简单的搜索命令可提供查询捷径：whatis 和 apropos。whatis 命令搜索大部分 Linux 数据库都有的命令摘要数据库，以寻找完整单词的匹配，如下所示。

```
$ whatis useradd
```

```
useradd(8) - create a new user or update default new user information
```

whatis 搜索返回了 useradd 命令并包含关于此命令功能的简短描述。

apropos 命令也搜索 whatis 数据库但是搜索字符串而不是完整的词。

```
$ apropos whoam
```

```
ldapwhoami(1) - LDAP who am i? tool
```

```
whoami(1) - print effective userid
```

apropos 搜索了 whatis 数据库，寻找 whoam 字符串并返回许多包含 whoam 的命令和操作。

还有一些有用的命令可以告诉我们命令的相关信息。例如 info 命令有时对一条命令的功能和选项提供更详尽的解释；尝试 info ls 以了解有关 ls 命令的更详细的信息。

3.5 用户与组

Linux 是多用户操作系统。这意味着它允许多个用户同时通过多个命令行或 GUI 会话连接。Linux 通过用户和组账号控制对主机及其资源的访问。另外，系统会为特定系统组件创

建用户以用于运行服务；例如，如果安装了一个邮件服务器，它会创建一个叫做 mail 的用户以用于此类服务，或者存在一个叫 lp 的用户（为行式打印机创建）控制打印机资源。

Linux 还依靠组。组是类似用户的集合。用户可以是一个组或多个组的成员，通常被放在某个组中以使其能够访问某类资源。例如需要访问 Accounts Payable 系统的所有用户可能要添加到一个叫做 accounts 的组。

■提示：用户和组的信息主要包含在两个文件里：/etc/passwd 保存用户信息，/etc/group 保存组信息。第 4 章将详细介绍这些文件。

用户和组是重要的概念，我们将在第 4 章介绍它们是如何工作的以及如何创建它们。概念上，用户和组的操作方式和在 Windows 主机上的操作方式相同。每个用户有一个账号，通常有一个密码保证安全。当大部分普通用户被创建时，一个类似于 Windows 配置档案的主目录也被创建。这个主目录给用户提供一个存储数据的地方，也是许多程序存储它们的用户自定义配置的默认位置。用户还属于一些组，正如在 Windows 中那样，这种机制为它们提供对辅助资源或服务的访问。

3.6 服务与进程

在 Windows 主机上，很多后台活动和服务器程序以服务的形式运行。服务可以启动和停止，通常必须在某个程序重新配置后重新启动。这些服务通常由控制面板上的服务管理器控制。Linux 主机上也有服务的概念。服务也叫后台程序（daemon），在主机上运行许多关键的功能。

与在 Windows 主机上的情况一样，每个服务或后台程序是运行在主机上的一个或多个进程。这些进程有名字；例如 Secure Shell 后台程序包括 master（后缀邮件服务器）、httpd（Apache 网络服务器）和 mysqld（MySQL 数据库服务器）。其中一些进程可能和许多执行各种系统和应用功能的其他进程一起默认运行。

在列表 3-4 中，我们使用带-A 标志（对所有的）的 ps 命令列出当前正在主机上运行的所有进程。

列表 3-4 ps 命令

```
$ ps -A
PID TTY      TIME CMD
1 ?          00:00:07 init
2 ?          00:00:10 migration/0
3 ?          00:00:00 ksoftirqd/0
4 ?          00:00:00 watchdog/0
5 ?          00:00:03 migration/1
6 ?          00:00:00 ksoftirqd/1
7 ?          00:00:00 watchdog/1
8 ?          00:00:00 events/0
9 ?          00:00:00 events/1
10 ?         00:00:00 khelper
11 ?         00:00:00 kthread
....
```

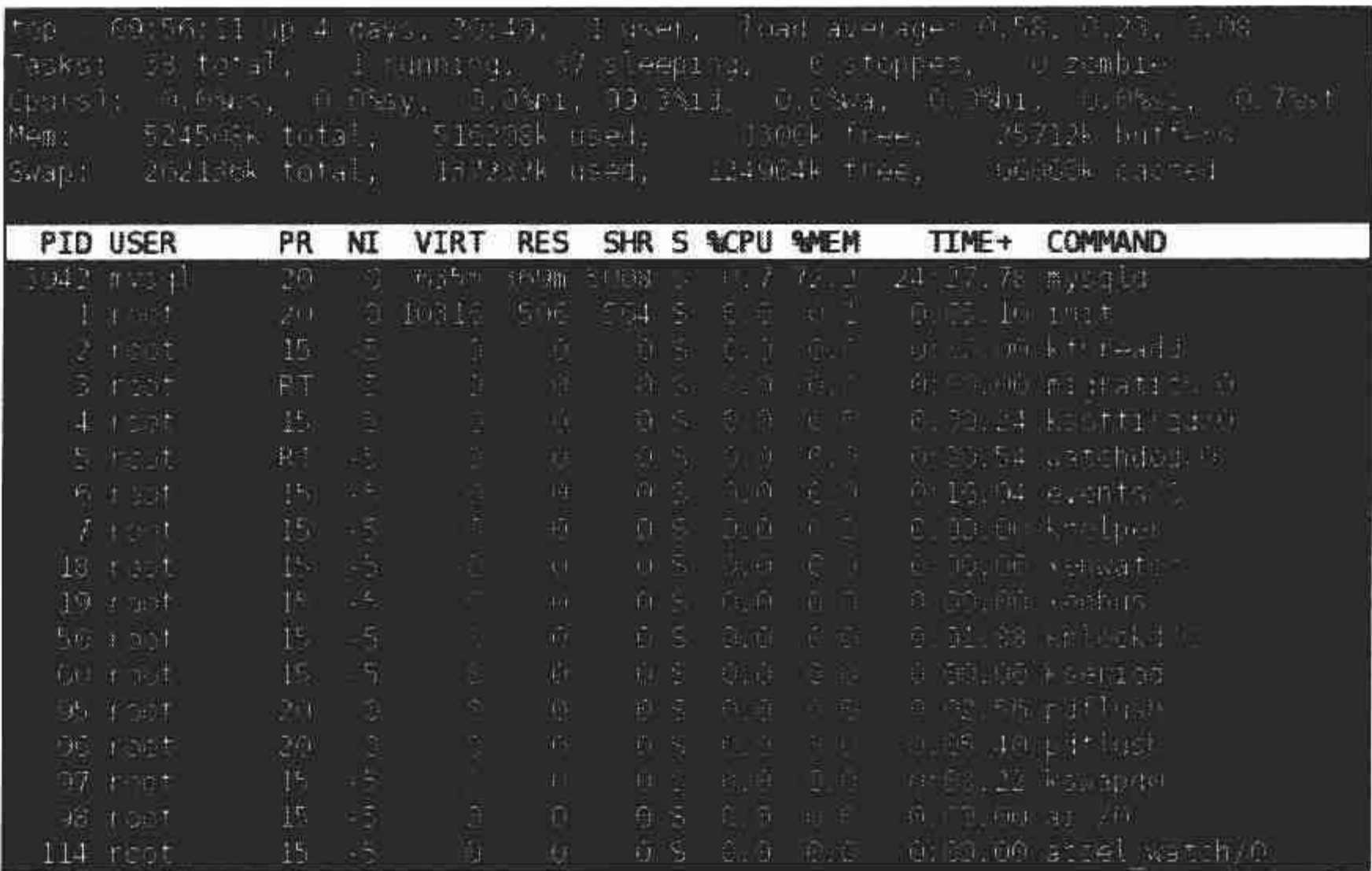

列表 3-4 显示了运行在主机上的进程列表。此表由带-A 选项的 ps 命令产生。运行的每个进程以进程 ID（Process ID，PID）的顺序列出，如左侧列所示。PID 用来控制进程，我们将在第 5 章讲述启动和停止进程时使用 PID。主机上最重要的进程叫做 init。Init（或初始化）进程是 Linux 主机的基本进程，它产生自身之外的所有进程。这个基本进程的 PID 总是 1，而且主机要正常工作就必须运行它。

另外还有一个有用的命令可以告诉我们哪些进程正在运行以及哪些进程消耗大部分的 CPU 和内存。这个命令叫 top，在列表 3-5 中运行了 top。

列表 3-5 top 命令

```
$ top
```

top 命令启动了一个交互的监视工具，它每几秒就对正在运行的进程更新一次。图 3-6 显示了 top 命令输出的结果。



```
top - 09:56:11 up 4 days, 20:49, 1 user, load average: 0.58, 0.25, 0.08
Tasks: 38 total, 1 running, 37 sleeping, 0 stopped, 0 zombie
Cpus:  0.6us,  0.0ms,  0.0%st,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.7%st
Mem:   524596k total,  516208k used,    8396k free,   25712k buffers
Swap:  262136k total,  167232k used,  114904k free,   66600k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1942	mysql	20	0	6356	36m	3108	S	11.7	72.1	24:27.76	mysqld
1	root	20	0	10416	506	504	S	0.0	0.1	0:05.10	init
2	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	smktthread
3	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	init
4	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	ksftttdm
5	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	watchdog
6	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	events
7	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	kernel
18	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	waiter
19	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	refers
50	root	15	-5	0	0	0	S	0.0	0.0	0:01.00	unlock
60	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	kernel
96	root	20	0	0	0	0	S	0.0	0.0	0:02.56	pdf
98	root	20	0	0	0	0	S	0.0	0.0	0:05.40	flash
99	root	15	-5	0	0	0	S	0.0	0.0	0:00.22	swap
98	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	at
114	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	at

图 3-6 top 进程监视命令

注：许多名字以 k 开头的进程并不是真正的进程，而是内核线程。这些线程是一种在操作系统核（即内核）里执行管理任务的特殊服务。如果主机运行了 KDE GUI，那么这些进程中有一些可能跟 KDE 相关。这些进程是可分辨的，因为内核进程总是以 root 用户身份运行，而 KDE 进程则很少是这样，除非是以 root 用户身份登录的 KDE 桌面。

3.7 软件包

在 Microsoft Windows 操作系统中，应用程序的安装一般是通过运行一个二进制程序，然后跟着一个安装过程来完成的。有些程序还带有卸载程序，卸载程序可以在不需要这些程序时移除它们。某些情况下，你可能要改为使用控制面板里的添加或删除程序的工具对程序进行添加或删除。

在 Linux 上，软件包管理器等同于添加或删除程序工具。一个软件包管理器包括一个事先打包好的程序集合，比如 Apache 网络服务器或者 OpenOffice 程序组件。这些事先打包好的程序被自然地称作软件包。打包的程序包括必需的二进制代码、支持文件，通常还有配置文件，安装之后可以直接运行。

第 7 章将详尽地介绍两种常用的包管理系统：RPM 和 Deb。它们分别由基于 Red Hat 和 Debian 的发行版使用。使用 Deb 的包括 Ubuntu、Debian 和其他许多发行版。

3.8 文件与文件系统

现在让我们看看 Linux 的文件和文件系统。下面从 pwd 命令或者说打印工作目录的命令开始。

```
$ pwd
/home/jsmith
```

pwd 命令通过确定工作目录或当前目录来定位用户在文件系统中的位置。从这里你可以浏览文件系统；首先使用 cd 或者叫更改目录命令把当前目录更改到根目录，如列表 3-6 所示。

列表 3-6 更改目录

```
$ cd /
/$
```

在列表 3-6 中，我们把当前目录更改为根目录/。根目录是目录树的基点。Linux 文件系统就是一个单独的目录树。这意味着 Linux 与 Windows 不一样，它只有一个层次化的目录结构。与多个驱动器，例如 C:\ 和 D:\，并且下面都有各自的目录树的情况不同，Linux 所有的驱动器、分区和存储都位于根目录/下。

这是怎么回事呢？当启动 Windows 主机时，它检测附带的驱动器并给它们分配驱动器号。相比之下，Linux 挂载驱动器和设备（这可以在启动时自动完成，也可以自己动手完成）。这些挂载的驱动器和设备在文件系统中以子目录的形式出现。

■注：第 8 章将介绍更多关于存储和挂载设备的内容。

使用 cd 命令你可以移动到其他目录和子目录。Linux 把这种在文件系统中来回移动所采取的步骤叫作一个路径。存在两类路径——绝对路径和相对路径。绝对路径总是以表示根目录的斜杠 (/) 开始，并指定要描述的确切位置。例如 /home/jsmith 就是一个绝对路径。

相对路径允许指定一个相对于当前位置或起始点的位置。例如命令

```
$ cd foobar
```

尝试把当前目录改到一个叫 foobar 的目录。如果这个目录不存在，cd 命令就无法完成。还有一对符号经常用于相对路径。

```
$ cd ..
```

.. 表示希望移动到目录树的上一级目录（如果已经在最上级了，那么就根本不会移至任何地方）。

我们还可以以其他方式使用这种机制在目录树中移动，如下所示。

```
$ cd ../foo/bar
```

在本例中，我们做了如下操作。

- 1. 如...符号表明的那样移动到上一级目录;
- 2. 在上一级目录更改到 foo 目录;
- 3. 然后在 foo 目录下更改到 bar 目录。

■注：如果习惯了 Microsoft Windows 命令行，你可能注意到隔开目录的是斜杠/而不是反斜杠\。需要花费一点时间来习惯它，但你很快就会适应的。

我们还可以使用下面的构造引用一个目录中的相对对象。

```
$ ./make
```

在这个命令前加上./将在当前目录执行 make 命令。

能够移动到哪些目录取决于你的权限。许多目录只允许特定用户和组访问（root 用户可以到任何位置）。如果尝试把目录更改到没有相应权限的地方，就会得到一个错误信息。

```
$ cd /root
-bash: cd: /root: Permission denied
```

■注：我们将在本章稍后的“权限”一节中介绍权限。

那么现在你知道了如何在目录树中来回移动。但是所有的东西都在主机的什么地方呢？大多数 Linux 发行版支持一个非常类似的目录结构。这不是说所有的发行版都相同，但是一般来讲，文件和目录都按照合乎逻辑又连贯一致的模式来安排。表 3-1 显示了 root 目录下典型的目录结构。每一项都包含对每个目录的简短说明。

表 3-1 Linux 目录结构

目录	说明
/bin/	用户命令和二进制文件
/boot/	引导程序使用的文件（第 5 章将介绍引导程序）
/dev/	设备文件
/etc/	系统配置文件
/home/	用户的主目录
/lib/	共享的库和内核模块
/media/	可移除媒介通常挂载到这里（参见第 8 章）
/mnt/	暂时挂载的文件系统通常挂载到这里（参见第 8 章）
/opt/	附加程序软件包
/proc/	内核和进程状态数据以文本文件的形式存储在这里
/root/	root 用户的主目录
/sbin/	系统二进制文件
/srv/	本主机提供的服务数据
/tmp/	临时文件的目录
/usr/	用户实用工具、库和应用程序
/var/	易变的或者临时文件和数据，比如日志、邮件队列和打印任务

■注：不是每个发行版都会有所有这些目录（可能还有另外的目录），但是这个列表通常还是准确的。

让我们看看表 3-1 中列出的根目录（/）下的一些关键目录。第一个也是最重要的目录之一是/etc/。/etc/目录以 *etcetera* 命名，主机上大部分重要的配置文件都位于其中。在往主机添加程序和服务时，你将频繁地与位于此目录中的文件打交道。

下面的/home/目录包括所有的用户主目录（除了 root 用户——它的主目录通常为/root/）。/tmp 目录通常是寻找临时文件的地方。你经常可以看到日志文件包含在/var/log/目录里，这些文件是由应用程序或通过主机的系统日志后台程序（syslog，或者 system logger）创建的。这些日志文件包含应用程序、后台程序和服务的状态相关的各种信息。

■提示：许多发行版也尝试标准化它们的目录结构以符合 Linux Standard Base (LSB)。LSB 是一个开放的标准，它试图为 Linux 操作系统提供标准。有关 LSB 的详细信息可以在 <http://www.linux-foundation.org/en/LSB> 上找到。

让我们仔细看一看这些文件和目录及其操作方式。先把目录更改到根目录/。

```
$ cd /
```

现在我们已经位于根目录下了，想看到此目录包含的内容，可以使用 ls 或列出目录、命令，如列表 3-7 所示。

列表 3-7 列出目录的内容

```
$ ls
bin dev etc lib lost+found mnt proc root sys usr
boot home lib64 media opt sbin srv tmp var
```

在列表 3-7 中，你可以看到 ls 命令返回了位于根目录的文件和目录列表。可见它看起来很接近表 3-1 中的列表。

默认情况下，ls 列出一个目录中的所有文件，但是可以限制它只显示一个文件名或者几个文件名，方法是在命令行里列出那个文件，如下所示。

```
$ ls foobar
```

这个命令会显示叫作 foobar 的所有文件或目录。我们还可以使用通配符或星号来挑选文件。

```
$ ls foo*
```

这将返回叫作 foo 和以 foo 开头的文件，比如 foobar，还有名字以 foo 开头的目录的内容。只指定星号会列出所有文件、目录和它们的内容。

■提示：你会看到更多的*符号，因为在 Linux 上使用它与在 Windows 上的情况相同。它表示一个通配符，用来代替一个或多个字母；例如刚才看到的 foo*，意思就是以 foo 开头的任何东西。单字母使用?符号匹配；例如，指定?at 会匹配 cat、mat 和 bat 等。这种活动都叫作 globbing，你可以到 <http://www.faqs.org/docs/abs/HTML/globbingref.html> 了解它在 Linux shell 中的用途。

你还可以通过指定目录名列出其他目录里的文件。

```
$ ls /usr/local/bin
```


这将列出/usr/local/bin 目录里的所有文件。

不过，在列表 3-7 中没有看到有关这些文件和目录的很多细节。它只显示了一个名单。要找到更多有关此列表的信息，你可以向 ls 命令后面追加开关以获得更多信息，如图 3-8 所示。

列表 3-8 从 ls 指令获得更多信息

```
$ ls -la
total 192
drwxr-xr-x 25 root root 4096 2008-07-22 12:47 .
drwxr-xr-x 25 root root 4096 2008-07-22 12:47 ..
-rw-r--r-- 1 root root      0 2008-07-15 20:47 .autofsck
drwxr-xr-x 2 root root 4096 2008-05-18 04:11 bin
drwxr-xr-x 6 root root 3072 2008-05-25 21:57 boot
drwxr-xr-x 14 root root 4100 2008-07-19 12:26 dev
drwxr-xr-x 116 root root 12288 2008-07-22 12:47 etc
drwxr-xr-x 7 smtpd smtpd 4096 2008-05-02 12:00 home
drwxr-xr-x 12 root root 4096 2008-05-17 18:14 lib
drwxr-xr-x 8 root root 4096 2008-06-06 10:19 lib64
drwx----- 2 root root 16384 2007-06-11 16:01 lost+found
drwxr-xr-x 2 root root 4096 2007-06-11 16:14 media
drwxr-xr-x 4 root root 4096 2007-06-12 11:28 mnt
...
```

在列表 3-8 中向 ls 命令追加了 l 和 a 开关。l 开关是 long 的缩写，使用了一个长的列表格式，正如所见，它显示了更多信息。a 开关告诉 ls 列出所有文件和目录，甚至包括隐藏的文件和目录。

提示：隐藏文件以句点为前缀（如列表 3-8 中的.autofsck 文件），经常用于保存配置和历史信息或者作为临时文件。

阅读 ls 命令的 man 页可以看到有关 ls 命令可用开关的完全列表——只要输入 man ls。

那么这个长列表格式表明了文件和目录的什么信息呢？在列表 3-8 中，每一项都返回一个有关它的小的信息集合。在列表 3-9 中可以看到该列表的一个子集，它显示了一个文件和一个目录，我们将仔细分析它。

列表 3-9 文件列表子集

```
-rw-r--r-- 1 root root      0 2008-07-15 20:47 .autofsck
drwxr-xr-x 2 root root 4096 2008-05-18 04:11 bin
```

列表的每一行包括每个对象的 7 项信息。

- UNIX 文件类型
- 权限
- 硬链接的数目
- 用户和组的所有权
- 大小
- 时间和日期
- 名字

列表中有些信息还引入了一些关键的 Linux 概念，比如权限、用户、组和所有权。我们将利用这次机会，不仅解释每一项，还要探究它们所代表的一些更宽泛的概念。

3.8.1 文件类型与权限

文件类型和权限包含在开始的 10 个字符中，即类似-rw-r--r--的那部分内容。这个好像很吓人的字母组合实际上很容易译码：第 1 个字母描述文件类型，接下来的 9 个字母描述文件的权限。

一、文件类型

几乎 Linux 文件系统上的所有东西一般都可以描述为文件。列表中的第 1 个字符说明到底是哪种文件。短划线（-）在这里表示一个可能包含数据和文本的常规文件，或者一个二进制可执行文件。d 表示一个目录，目录本质上是列出其他文件的文件。l 表示一个符号链接。符号链接可使文件和目录在文件系统的多个位置可见，很像 Microsoft Windows 中的快捷方式。

表 3-2 列出了可用的文件类型。

表 3-2 文件类型	
类型	说明
-	文件
d	目录
l	链接
c	字符设备
b	块设备
s	套接字（socket）
p	有名管道（named pipe）

在这里简短地介绍一下其他类型。其中的大多数类型通常都不会用到，但偶尔会在后面的章节中出现。b 和 c 文件类型用于不同类型的输入输出设备（如果看看/dev 目录，就可以找到这些设备文件的例子）。设备使得操作系统可以与特定的硬件设备交互；例如，许多发行版都有一个叫/dev/dvd 的设备，它表示附带在主机上的 DVD 驱动器。

■提示：在第 8 章说明如何在主机上加载 CD 或 DVD 时，你将会学到更多有关设备的知识。

最后，套接字和有名管道是可达成各类进程间通信的文件。它们使进程得以相互通信。在本书后面可以看到一些套接字和有名管道的例子。

二、权限

接下来的 9 个字符详述了分配给这个文件或目录的访问权限。在 Linux 里，权限用来确定用户和组对文件可以进行何种类型的访问。控制对文件和程序的权限和访问对 Linux 主机的安全很关键，本书经常使用权限来提供适当的文件访问。因此理解权限的工作方式以及如何更改权限是一件重要的事情。

文件有 3 种常规权限类型。

- 读权限，用字母 r 表示。
- 写权限，用字母 w 表示。

- 执行权限，用字母 x 表示。

■注：还有另外两种权限类型：sticky 和 setuid/setgid 权限，分别用 t 和 s 表示。本章后面的补充知识“Setuid、Setgid 与 Sticky 权限”将介绍它们。

读权限允许阅读或查看文件，但不可以编辑文件。如果对目录设置读权限，那么该目录内的文件名是可读的，但其他的细节，如文件的权限和大小，都不可见。写权限允许对文件做修改或写入操作。如果对目录设置写权限，那么就可以创建、删除和重命名目录内的文件。执行权限允许运行文件；例如，所有二进制文件和命令（二进制文件与 Windows 可执行文件相似）必须标记为可执行才允许运行。如果该权限设置对象为目录，那么就可以遍历该目录，比如通过 cd 命令访问一个子目录。因此，对目录设置读权限和执行权限的组合就可以遍历该目录并查看它的内容细节。

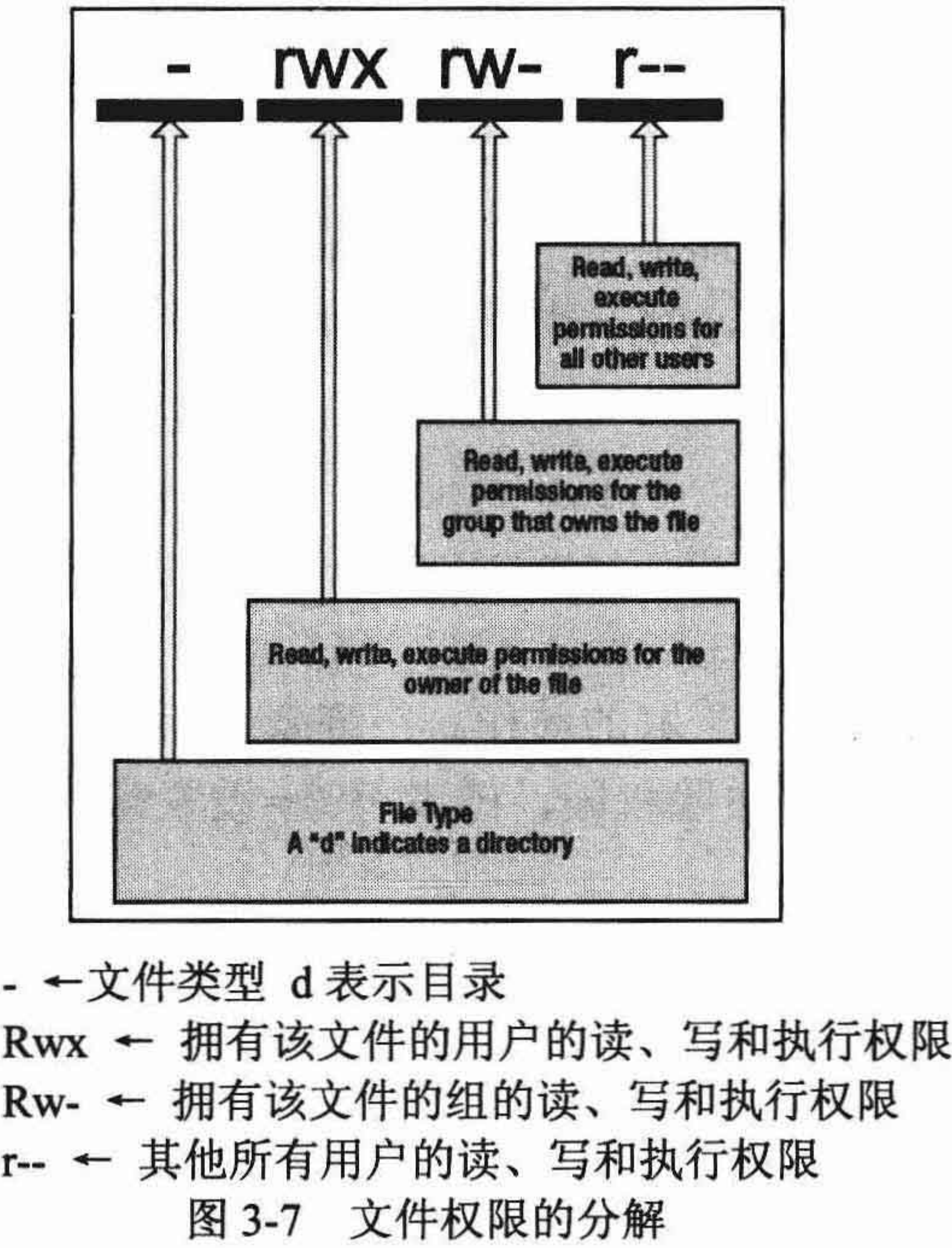
主机上的每个文件都有 3 类权限。

- 用户类。
- 组类。
- 其他（其他任何人）类。

每一类都代表对文件不同类别的访问。用户类描述拥有文件的用户的权限。这是文件列表中的头 3 个字母。组类描述拥有文件的组的权限。这是接下来的 3 个字母。

■注：Linux 里的组是用户的集合。组使相似的用户共同拥有对程序和服务的权限；例如，Accounting 部门的所有用户可属于同一个组，以使它们可以访问 Accounts Payable 程序。第 4 章将会介绍组。

最后，其他类描述其他所有人对文件所拥有的权限。这是文件列表中的最后 3 个字母。
图 3-7 描述了这些类及其位置。



■注：任何位置的短划线都意味着特定权限根本没有设置。

列表 3-10 显示了一个文件，我们将仔细分析它的权限，然后学习如何对这些权限进行修改。

列表 3-10 权限

```
-rw-r--r-- 1 root root      0 2008-07-15 20:47 myfile
```

列表 3-10 显示的是一个文件，开头的短划线说明了这一点。该文件归 root 用户和 root 组所有。头 3 个权限是 rw-，表明 root 用户可以读写该文件，但是短划线意味着没有设置执行权限，因此该文件不能被用户执行。接下来的 3 个权限是 r--，表明任何属于 root 组的用户都可以读该文件但不能做其他任何事情。最后 3 个权限为 r--，表示其他类用户所拥有的权限。在本例中，其他的人可以读该文件但不能写或执行。

现在已经了解了权限的意思，但如何修改权限呢？权限通过 chmod（修改文件的模式比特）命令修改。修改权限的关键是只有拥有该文件的用户或者 root 用户可以修改文件权限。因此，在列表 3-10 中，只有 root 用户可以修改 myfile 文件的权限。

chmod 命令的语法很简单。从列表 3-11 可以看到一些权限改动的例子。

列表 3-11 修改权限

```
# chmod u+x myfile
# chmod u-x, og+w myfile
# chmod 654 myfile
```

列表 3-11 对 myfile 文件的权限修改了 3 次。权限修改通过指定类、要执行的行动、权限本身以及想要修改的文件来完成。在第一个例子中可以看到 u+x。它的意思是为用户类增加执行权限。

■注：执行权限通常只在如脚本或二进制文件（又叫应用程序或程序）这样本身可执行的文件或目录上设置。

更新之后，文件权限如下所示。

```
-rwxr--r-- 1 root root      0 2008-07-15 20:47 myfile
```

你可以看到用户类权限中增加了 x。那么 chmod 是如何知道要那么做呢？对，修改命令中的 u 代表用户类。对于 chmod，每一类都缩写为一个字母。

- u: 用户 (User)。
- g: 组 (Group)。
- o: 其他人或每个人 (Other or everyone)。
- a: 所有类 (All)。

在类后面要指定希望对该类采取的动作。在列表 3-11 中的第一行，符号+表示增加一个权限。可以指定符号-从某类中去除权限，或者符号=设置绝对的权限。最后要指定执行的权限，本例是 x。

你还可以在单个命令中指定多个权限改动，如列表 3-11 第二行所示。这一行的改动是 u-x 和 go+w。这会从用户类移除 x 或叫执行权限，并往组类和其他类增加 w 或叫写权限。你可以看到，权限改动之间使用逗号隔开，并且可以列出多个要修改权限的类。（你还可以列出多个权限；例如 u+rw 可以往用户类增加读写权限。）

因此列表 3-11 中的第二行会使文件权限变成下面这样。

```
-rw-rw-rw- 1 root root 0 2008-07-15 20:47 myfile
```

对于 `chmod`，你还可以使用类的缩写 `a`，表明某个动作应用于所有的类；例如 `a+r` 会把读权限添加到所有类：用户类、组类和其他类。

我们还可以使用符号 `=` 把一个类的权限应用到另一个类。

```
# chmod u=g myfile
```

在上一行，用户类的权限被设置得与组类相同。

你还可以通过列出多个由空格分开的文件来为其设置权限，就像下面这样。

```
# chmod u+r file1 file2 file3
```

正如 `ls` 命令那样，你还可以引用其他位置的文件，就像下面这样。

```
# chmod u+x /usr/local/bin/foobar
```

上一行为位于 `/usr/local/bin` 目录下的 `foobar` 文件添加了用户类的执行权限。

你还可以使用星号指定所有的文件并增加 `-R` 开关来递归查找下一级目录，就像下面这样。

```
# chmod -R u+x /usr/local/bin/*
```

上一行的 `chmod` 命令为 `/usr/local/bin` 目录里的每个文件添加了用户类的执行权限。

列表 3-11 中的最后一行有点不同。它没有指定类和权限，而是指定一个号码 `654`。这个号码叫做八进位计数。每个数字代表 3 个类中的一个：用户类、组类或其他类。而且每个数字是分配给那个类的权限之和。从表 3-3 可以看到分配给每个权限类型的值。

表 3-3 八进制权限值

权限	值	描述
r	4	读
w	2	写
x	1	执行

每个权限值加在一起，就为每个类生成了范围为 1~7 的结果。因此，列表 3-11 中的值 `654` 代表这些权限。

```
-rw-r-xr-- 1 root root 0 2008-08-14 22:37 myfile
```

第一个值是 `6`，等于为用户类分配了读权限 `4` 和写权限 `2`。第二个值是 `5`，代表分配给组类读权限 `4` 和执行权限 `1`。最后的值是 `4`，则表示只分配给其他类读权限。为了弄得更清楚，可通过表 3-4 了解从 0~7 所有可能值的列表。

表 3-4 八进制值

八进制数字	权限	说明
0	---	无权限
1	--x	执行
2	-w-	写

续表

八进制数字	权限	说明
3	-wx	写与执行
4	r--	读
5	r-x	读与执行
6	rw-	读与写
7	rwx	读、写与执行

表 3-5 给出了一些常用的八进制数及其代表的权限。

表 3-5 八进制权限

八进制数	权限
600	rw-r--r--
644	rw-r--r--
664	rw-rw-r--
666	rw-rw-rw-
755	rxwxr-xr-x
777	rxwxrwxrwx

■提示：chmod 命令还有一些其他修改权限的语法，可以到该命令的 man 页了解。

最后，有一个重要的概念叫作 umask，同样需要理解它以便充分领会权限的工作方式。umask 规定了一个文件创建时的默认权限设置。一般地，不用 umask 设置的话，文件被创建时的权限是 0666（或者说对拥有者、组和其他用户都设置为读写权限），目录被创建时的权限是 0777（或者说对所有所有者、组和其他用户都有读写和执行的权限）。你可以使用 umask 命令修改这些缺省权限。让我们看一个例子。

```
# umask 0022
```

这里指定 umask 为 0022。这看起来很熟悉，不是吗？是的，它是一种八进制计数法；在本例中，它表示哪些不被授权。那么，在这里用默认的权限，例如 0666，减去 0022，留下的就是 0644 的权限。使用了 0022 的 umask 之后，新文件被创建时对拥有者有读写的权限，对组和其他用户只有读权限。新创建的目录有 0755 的权限。另一个常用的 umask 是 0002，这样对文件的默认权限是 0664，对目录的默认权限是 775。它允许组类用户也有写权限，经常用于位于共享目录中的文件或文件共享。

大部分主机上的 umask 都由 shell 自动设置。对于 Bash shell，一般可以在/etc/bashrc 文件中找到全局 umask，但也可以基于单个用户使用 umask 命令替换它。

■提示：umask 命令还可以利用另外的语法设置 umask。刚才已经描述了最简单和最容易的一种，更多的细节可以在 umask 的 man 页找到。

补充知识：SETUID、SETGID 和 STICKY 权限

还有另外两种需要重点理解的权限类型：setuid/setgid 和 sticky。

setuid 和 setgid 权限使得用户运行命令就好像他就是拥有该命令的用户或组一样。那么为什么需要这种权限呢？因为这使得用户可以执行它们正常情况下会被限制执行的特殊任务。

passwd 命令就是一个很好的例子。passwd 命令允许用户修改他的密码。要做到这一点，该命令需要对访问受限的密码文件执行写操作。通过增加 setuid 权限，用户可以执行 passwd 命令并且可以像 root 用户一样运行它，从而允许他更改密码。

通过权限列表中的 s 或 S，你可以识别 setuid 和 setgid 权限。例如，passwd 命令的权限如下所示。

```
-rwsr-xr-x 1 root root 25708 2007-09-25 17:32 /usr/bin/passwd
```

在用户类的执行权限位置可以看到 s，表明 passwd 命令设置了 setuid 权限。在大多数发行版中，setuid/setgid 权限被吝惜地用于许可这种访问。这些权限很少使用的原因是，你通常不希望一个用户能和另一个用户一样运行程序或者拥有特别高的特权（拥有特权的另外一种方式是使用 su 和 sudo 命令，这将在第 4 章进一步说明）。因为它们还可能被滥用，并且有安全泄漏的问题，所以不应该被大范围地使用。本书中可能会看到一两个应用程序使用了 setuid/setgid 权限。

sticky 权限稍微有点不同，它用于目录（对文件不起作用）。如果对某个目录设置了 sticky 位，那么该目录里的文件只能被拥有它的用户或 root 用户删除，不管该目录的其他权限设置如何。这样可以创建公共目录。在这种目录下，每个用户都可以创建文件但只能删除它们自己的文件。通过其他类的执行权限位置上的 t 可以识别拥有 sticky 权限的目录。它最常用来设置/tmp 目录。

```
drwxrwxrwt 4 root root 4096 2008-08-15 03:10 tmp
```

在八进制计数法中，setuid/setgid 和 sticky 权限由在八进制数之前的第 4 个数字表示，如 6755。与其他权限一样，每个特定的权限也有一个数字值：setuid 为 4，setgid 为 2，sticky 为 1。于是，为了设置某个目录的 sticky 位，要使用像 1755 这样的八进制数。如果没有设置 setuid/setgid 或 sticky 权限，那么前面的数字就是 0，如下所示。

```
# chmod 0644 /etc/grub.conf
```

3.8.2 链接

让我们看看列表 3-9 中的另外一个例子。

```
-rw-r--r-- 1 root root 0 2008-07-15 20:47 .autofsck
drwxr-xr-x 2 root root 4096 2008-05-18 04:11 bin
```

在该列表中，文件类型和权限之后是该文件硬链接的数目。硬链接是把文件连接到存储卷物理数据的引用。对特定的数据可以有多个链接。不过，硬链接不同于较早前介绍的符号链接（通过使用 l 文件类型表示），尽管两者由同一个命令 ln 创建。本章后面的“链接文件”一节将介绍 ln 命令。

3.8.3 用户、组与所有权

列表中接下来是文件的所有权。每个对象都由一个用户或者组拥有；列表 3-9 中的对象

由 root 用户或者 root 组所有。在介绍权限时简单地讲到过用户和组，讲到只有拥有文件的用户可以修改它的权限，而且组是用户的集合。组一般用于对资源的访问许可；例如，所有需要访问某个打印机或文件共享功能的用户可以属于提供对这些资源访问的组。正如本章较早前所讲的那样，Linux 主机上的一个用户必须至少属于一个组，即基本组（primary group），但是它还可以属于一个或更多另外的组，叫做附加组（supplementary group）。

可以使用 `chown` 命令更改文件的用户和组所有权。只有 root 用户有权力更改文件的用户所有权（尽管可以利用在本章较早前介绍过并将在第 4 章详细介绍的 `sudo` 命令来获得这种权力）。

列表 3-12 给出了一些例子，说明如何使用 `chown` 命令改变用户和组所有权。

列表 3-12 更改所有权

```
# chown jsmith myfile
# chown jsmith:admin myfile
# chown -R jsmith:admin /home/jsmith/*
```

列表 3-12 中有 3 个 `chown` 命令。第一个命令把拥有 `myfile` 文件的用户更改为 `jsmith`。第 2 个命令更改了该文件的用户和组所有权，拥有者的用户变为 `jsmith`，组变为 `admin`。所有者和组用冒号:分开。第三个也是最后一个命令使用 `-R` 开关打开了递归功能。该命令可以把 `/home/jsmith` 目录下的所有文件和目录的所有者更改为用户 `jsmith` 和组 `admin`。

■注：我们还可以使用 `chgrp` 命令。它允许一个无特权用户更改一个文件的组。该用户只能把组所有权更改到该用户所在的组。像这样使用它：`chgrp groupname file`。

3.8.4 大小和空间

在列表中接下来看到的是这个对象在磁盘上的大小。文件的大小以字节为单位（1000 字节是 1 千字节，或 KB）。通过添加 `-h` 开关还可以用人可读的格式显示大小，如下所示。

```
$ ls -lh
-rw-rw-r-- 1 jsmith jsmith 51K 2008-08-17 23:47 myfile
```

在上一行里，你可以看到 `myfile` 文件大小为 51KB。

在一个列表中，挨着目录的大小不是它的总大小而是目录元数据的大小。要得到某个目录中所有文件的总大小，可以使用 `du` 或叫磁盘使用率命令。指定（或更改到）希望找出其总大小的目录并运行带 `-s` 和 `-h` 开关的 `du` 命令。`-s` 开关计算总大小，而 `-h` 开关以人可读的形式显示出来。

```
$ du -sh /usr/local/bin
4.7M /usr/local/bin
```

`du` 工具有许多开关和选项，可以到它的 `man` 页了解。

除了文件和目录的大小，你还可以使用另外一个命令 `df` 查看主机上总的已用和可用磁盘空间。该命令显示所有的磁盘和存储设备及其当前空闲空间。列表 3-13 显示了 `df` 命令。

列表 3-13 显示磁盘空间

```
$ df -h
Filesystem                                Size Used Avail Use% Mounted on
/dev/mapper/VolGroup00-LogVol01          178G 11G  159G   6%  /
/dev/sda1                                99M  37M   58M  39%  /boot
tmpfs                                     910M  0    910M   0%  /dev/shm
```

我们执行了该命令，并添加了-h 开关以使它返回人可读的大小。它除了显示使用百分比，还显示当前的文件系统和已使用及未使用的空间。df 命令还有其他选项，可以到该命令的 man 页了解。第 8 章将返回来介绍 df 和 du 命令。

3.8.5 日期与时间

列表中倒数第二项和最后一项是该文件上次修改的日期和时间（也叫 mtime）以及文件和目录名。Linux 还记录文件上次被访问的时间（叫作 atime）以及它的创建时间（叫作 ctime）。可以通过使用-u 开关列出一个文件来显示它上次被访问的时间，如下所示。

```
$ ls -lu
你可以使用-c 开关列出创建日期，如下所示。
$ ls -lc
```

■注：第 17 章将返回来介绍 atime。

如果想知道当前主机的实际时间和日期，可以使用有效而强大的 date 命令。不带任何选项地使用 date 命令将返回当前的时间和日期，如下所示。

```
$ date
Tue Aug 19 13:01:20 EST 2008
```

还可以在 date 命令上添加开关形成不同日期或时间格式的输出；例如，为了显示 Unix 纪元时间（从 1970 年 1 月 1 日开始算的秒数），可以如下所示执行 date 命令。

```
$ date +%s
1219116319
```

这里使用+符号来添加一种格式，然后指定格式。本例中指定了%s 以显示 Unix 纪元时间。其他格式可以到 date 命令的 man 页查看。使用 date 命令还可以设置时间。键入 date，然后以 MMDDHHMM[[CC]YY]的格式指定所需的日期和时间。有关 Unix 纪元时间的更多信息可以到http://en.wikipedia.org/wiki/Unix_time查阅。

■注：这只是设置时间的一种方式，第 9 章将介绍其他更有效的方式，如 NTP。

3.9 文件操作

那么在探究简单的文件列表的课程里，已经涉及了很多概念，介绍了一些 Linux 命令，

并讲述了如何执行几个关键的管理任务。从这些任务继续下去，我们将以介绍如何查看、编辑、搜索、复制、移动和删除文件结束本章。你应该知道如何处理所有这些任务以便管理 Linux 主机。

3.9.1 读文件

你要学习的第一件事情是如何读文件。Linux 上的许多文件，特别是配置文件，都是基于文本的，可以使用简单的命令行工具阅读。

■注：始终要记得，要读某个文件，必须有对该文件的读权限。这意味着你需要拥有它或者属于对该文件有读权限的组，再或者该文件对其他类用户设置了读权限。

第一个这样的工具是 `cat`。`cat` 命令之所以这样命名，是因为它“串联并打印文件”。列表 3-14 显示了 `cat` 命令在文本文件上的使用。

列表 3-14 使用 `cat` 命令

```
$ cat /etc/hosts
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1          localhost.localdomain localhost localhost
::1               localhost6.localdomain6 localhost6
```

列表 3-14 把 `/etc/hosts` 文件输出到屏幕。`/etc/hosts` 文件包含了 Linux 主机的主机输入项（就像 Windows 里的 `\Windows\system32\services\etc\hosts` 文件一样），这些输入项匹配主机名与 IP 地址。但是 `cat` 是一个相当简单的工具，只是直接输出文本。如果文件很大，文本会一直输出并翻页，就是说，如果想看文件开头的部分，需要翻回来。

■提示：可以使用组合键 `Shift+Page Up` 和 `Shift+Page Down` 往上和往下翻阅虚拟控制台。

为了克服这个问题，我们将考虑另外一个命令 `less`。

■注：你可以尝试对 `/etc/passwd` 和 `/etc/group` 文件使用 `cat` 命令，以查看主机上用户和组的清单。

使用 `less` 命令可以往前或往后翻阅文件，一次一屏。每次显示一页时，都会有希望如何继续的提示。通过指定文件名运行 `less`，如下所示。

```
$ less /etc/services
```

从 `less` 界面内可以翻阅整个文件。要到下一页，使用空格键。要每次前进一行，使用 `Enter` 键。要往后翻，使用 `B` 键。还可以使用箭头键翻阅。要退出 `less` 命令，使用 `Q` 键。

■注：还有其他使用 `less` 浏览文件的方式，你可以阅读该命令的 `man` 页了解。

除了浏览文件，你还可能会搜索一个或多个文件来寻找特定的信息。为此，你可以利用十分强大的 `grep` 命令。使用 `grep` 命令可以在一个文件或多个文件中搜索一个字符串或样式

(使用正则表达式) 并返回搜索的结果。

■注：单词 `grep` 在 IT 界已经成为一个常用的搜索术语，十分类似在线搜索引擎领域的术语 `google`。2003 年，牛津英语词典增加了既可作名词又可作动词的单词 `grep`（例如，“John `grep`'ed his mailbox to find the e-mail”）。

如列表 3-15 所示，在文件 `/etc/hosts` 中对字符串 `localhost` 做了一个非常简单的 `grep` 搜索。

列表 3-15 引入 `grep`

```
$ grep localhost /etc/hosts
127.0.0.1          localhost.localdomain localhost localhost
::1               localhost6.localdomain6 localhost6
```

要使用 `grep`，你需指定要搜索的字符串，本例中就是 `localhost`（`grep` 对大小写敏感，所以它只寻找这个小写的字符串），然后是搜索的文件名。

■注：通过添加 `-i` 开关可以使 `grep` 对大小写不敏感。

默认情况下，`grep` 返回该文件中包含预定字符串的那些行。还可以使用星号搜索一个以上的文件，正如本章较早前在别的命令中演示的那样，如下所示。

```
$ grep localhost /etc/host*
$ grep localhost /etc/*
```

第一个命令会搜索 `/etc/` 目录中所有以 `host*` 开头的文件，第二个命令会搜索 `/etc/` 目录中所有的文件。两个搜索都针对字符串 `localhost`。

还可以添加 `-r` 开关递归搜索下一级目录，如下所示。

```
$ grep -r localhost /etc
```

■提示：在 Ubuntu 和 Debian 主机上，`rgrep` 命令自动递归搜索目录。

还可以指定更复杂的搜索项，如多个词，如下所示。

```
$ grep "local host" /etc/hosts
```

你可以看到这里指定了 `local` 和 `host`，用一个空格隔开了两者。为了告诉 `grep` 这些单词是一起的以使之得到正确解析，要用引号引起来。许多命令通常都在命令中使用引号以避免输入被不正确地解析。本例精确搜索字符串 “`local host`”，而 `grep` 没有返回结果，因为这个字符串在 `/etc/hosts` 文件中不存在。

`grep` 命令能够处理比这些简单的例子更复杂的情况。你可以使用 `grep` 在文件中做复杂的正则表达式搜索，如下所示。

```
$ grep 'J[oO][bB]' *
```

它会在当前目录下的所有文件中寻找字符串 `JOB`、`Job`、`JOB` 或者 `JoB`（记住，`grep` 默认对大小写敏感，因此该正则表达式明确指定了大小写的变化）。使用正则表达式可以在主机里做一些很强大的搜索。

让我们用 `grep` 做另外一些有用的正则表达式搜索。

```
$ grep 'job$' *
```

上一行搜索当前目录下的所有文件，以寻找以 `job` 结束的字符串。`$` 符号告诉 `grep` 在字符串末尾搜索该文本。

相反地，可以用 `^` 符号搜索以特定字符串开头的字符串，如下所示。

```
$ grep '^job' *
```

它会返回所有以 `job` 开头的字符串。你会发现无数的很有用的经常使用的其他的正则表达式。

■注：正则表达式，或称 `regexes`，是一种用来识别文本字符串的形式化语言；例如，一个正则表达式可以识别一个文件中对字符串 `job` 的全部引用。各种各样很相似的正则表达式语言被像 `grep` 这样的工具和像 Perl 这样的编程语言所使用。大多数正则表达式语言的语法很相似，但有时有些细微的差别。你可以到 http://en.wikipedia.org/wiki/Regular_expression 深入了解正则表达式。我们还推荐阅读如 Jeffrey Friedl 的《精通正则表达式》(Mastering Regular Expressions, O'Reilly, 2006) 这样的书以帮助学习正则表达式。

3.9.2 搜索文件

我们已经介绍了如何读文件，但如果要查找某个文件的位置该如何做？Linux 上的许多命令和工具都可用来寻找文件，其工作方式与 Windows 的搜索功能非常相似。图 3-8 显示了 Gnome 的搜索功能。



图 3-8 Gnome 搜索功能

在命令行上，我们还可以使用 `find` 命令搜索文件。让我们使用 `find` 命令在 `/home` 目录里搜索一个叫作 `myfile` 的文件。


```
$ find /home/ -type f -iname myfile*
```

find 命令的使用方式很简单。首先指定要搜索的位置，本例中就是/home 目录。对 root 还可以指定/（这样就搜索整个目录树），或者其他任何可以访问的位置。

■注：如果没有权限搜索某个目录，将得到一个表示搜索被拒绝的错误信息。

接下来还指定了两个选项，**-type** 和**-iname**。第一个选项**-type** 指明要搜索的文件类型；本例中用 **f** 表示是一个常规文件。还可以用 **d** 表示目录或者 **s** 表示套接字（可到 man 页查找所有可能的类型）。**-iname** 选项搜索大小写不敏感的式样，本例中，所有文件以 **myfile** 开始。这些选项只是可能的搜索选项中极小的部分；还可以通过拥有者、组、权限、创建或修改的日期和时间、大小等搜索。然后 **find** 命令会搜索指定的位置并返回与搜索标准相符的文件清单。

find 命令还可以用来定位那些不为任何用户和组所有的文件和目录。这经常发生在这种情况下：某个用户或组已经被删除，而与该用户或该组相关联的文件没有重新分配或移除。第 4 章将对此进行详细的讨论。使用下面的 **find** 命令可以列出这种状态下的全部文件。

```
# find / -nouser -o -nogroup
```

这个以 root 用户身份运行的命令将搜索整个目录树，以寻找所有不属于有效用户或有效组的文件。

■提示：有一些其他相关的搜索命令你也许希望了解，包括 **locate**、**whereis** 和 **which**。可以阅读它们的 man 页以得到更多信息。

3.9.3 复制文件

除了查看文件之外，管理主机时最常采取的行动之一就是复制文件。理解复制文件的第一件事就是，与读文件一样，需要有适当的权限才能复制。要复制一个文件，你需要两个权限：要复制的文件的读权限和复制目的地的写权限。

要复制文件，请使用 **cp** 命令（**copy** 的简写）。列表 3-16 显示了一个简单的 **cp** 命令。

列表 3-16 复制文件

```
$ cp /home/jsmith/myfile /home/jsmith/yourfile
```

列表 3-16 把文件/home/jsmith/myfile 复制到/home/jsmith/yourfile。在使用 **cp** 命令的时候要仔细一点。默认情况下，**cp** 命令会不提示就覆盖已存在的文件。如果已经有一个文件和要复制的文件同名，这可能不好。你可以通过添加**-i** 开关来改变这种行为。**-i** 开关打开交互模式，如果正在复制的文件已经存在，就会得到一个是或否的问题提示。回答 **y** 是覆盖，**n** 是退出复制。

■注：在 Red Hat、Fedora 和 CentOS 里，**cp**、**mv** 和 **rm** 的**-i** 开关通过为每个命令命名别名而得以自动设置；例如 **cp -i** 的命令别名是 **cp**。在其他发行版上，你可以使用 **alias** 命令做到这一点；更多细节见<http://www.ss64.com/bash/alias.html>。

如果没有文件的读权限，将得到这样的错误信息。

```
cp: cannot open `/home/jsmith/myfile' for reading: Permission denied
```

如果不能写到目标位置，将得到一个类似的错误信息。

```
cp: cannot stat `/home/jsmith/yourfile': Permission denied
```

使用 **cp** 还可以做更多的事情。可以使用星号复制多个文件，如下所示。

```
$ cp /home/jsmith/* /home/jsmith/backup
```

上一行的目标 `/home/jsmith/backup` 必须是个目录，而现在正在把 `/home/jsmith` 里的所有文件复制到这个目录。

还可以选择文件的子集。

```
$ cp -i /home/jsmith/*.c ./
```

上一行把所有后缀为 `.c` 的文件复制到当前目录（使用了 `./` 快捷方式）。还添加了 `-i` 开关以确保在文件已存在时给予提示。

你还可以使用带 `-r` 开关的 **cp** 命令复制目录和它的内容。

```
$ cp -r /home/jsmith /backup
```

上一行命令把 `/home/jsmith` 目录和它下面的所有文件和目录复制到 `/backup` 目录。

■注：与在 Windows 上一样，当你使用 `-r` 开关时，小心不要使用 `.*` 通配符。当在 Linux 上使用时，`..` 目录也会被以递归的方式复制，这可能不是你的意图！

最后，当使用 **cp** 命令复制文件时，文件的一些选项，如日期、时间和权限，可能被修改或更新。如果在复制时想保留原始值，可以使用 `-p` 开关。

```
$ cp -p /home/jsmith/myfile /home/jsmith/yourfile
```

补充知识：目录操作

除了文件，你还可以对目录进行操作。你要创建目录，就使用 **mkdir** 命令。你必须拥有目录创建所在位置的写权限。如果想复制目录并递归地复制它的内容，你可以使用带 `-r` 开关的 **cp** 命令来完成。

你还可以使用 **mv** 命令移动目录，与移动文件的方式相同。

最后，你如果想删除目录，就使用 **rmdir** 命令。**rmdir** 命令只能移除空目录（即里面没有文件的目录）。

较早前分析过的 **cat** 命令通过利用一个叫重定向的命令行功能也可以用来复制文件。

```
$ cat /home/jsmith/myfile > /home/jsmith/yourfile
```

`>` 符号的用途是把一个命令的输出发送到 `>` 符号另一边的命令或动作上。本例中，**cat** 命令的输出被重定向到一个叫 `yourfile` 的文件。如果此文件不存在，它将被创建。如果存在，它的内容会被改写。

■注：使用重定向时要小心，因为目标文件会在没有警告的情况下被覆盖。

还可以使用同样的机制追加文件。

```
$ cat /home/jsmith/myfile >> /home/jsmith/yourfile
```

使用>>语法会把 myfile 的 cat 输出追加到 yourfile 的末尾。如果 yourfile 不存在,就创建它。

提示: 重定向也可以被许多其他命令用来把一个命令的输出定向到另一个命令。这也和另外一个叫作管道 (piping) 的功能密切关联 (见补充知识“管道与其他 Bash 提示和技巧”)。

补充知识: 管道与其他 Bash 提示和技巧

我们已经快速浏览了 Bash 命令行,了解了一些可做的事。这包括使用带>或>>的重定向功能把一个命令的输出重定向到另一个命令。这个概念可以用|或叫管道符号扩展。管道把一个命令的输出传递给另一个命令,如下所示。

```
$ cat /etc/passwd | grep ataylor
```

上一行输出了/etc/passwd 文件的内容,然后把结果传递给 grep 命令以搜索 ataylor。这样就会输出所有包含 ataylor 的行。在命令行上几乎可以对任何接受输入的命令这样做。一些可用于管道的有用命令是 sort (按各种方式对输入排序)、uniq (生成条目不重复的列表) 和 wc (计行数、字数等)。可以到这些命令的 man 页了解它们。

你还可以进一步利用重定向,定向多次或者同时使用管道和重定向。让我们看一个例子。

```
$ cat *.txt | sort | uniq > text
```

这个例子要求主机输出所有后缀为.txt 的文件,按字母顺序为它们排序,删除重复的行 (使用 uniq 命令),然后把结果输出到一个叫 text 的文件 (该文件如果不存在就创建一个,如果存在就覆盖之)。

除了输出,输入也可以重定向。

```
$ grep accounts < /etc/group > matched_accounts
```

这个例子使用<符号把文件/etc/group 定向到 grep 命令。然后告诉 grep 搜索 accounts,并使用>符号把该命令的输出定向到一个叫 matched_accounts 的文件。

另外一个有用的技巧就是在一个命令行中运行多个由分号隔开的命令。

```
$ ./configure; make; make test
```

这行命令会运行当前目录里的 configure 脚本,然后是 make,最后是 make test 命令。这些命令会按顺序逐一运行。

这些只是展示 Bash 命令行、重定向和管道功能的非常简单的例子。要了解更多的 Bash 功能,可以查阅 Bash 的 man 页 (man bash),或者查看 Bash 在线指南之一,如 http://www.hypexr.org/bash_tutorial.php、<http://tldp.org/LDP/Bash-Beginners-Guide/html/>,或者到 <http://www.faqs.org/docs/bashman/bashref.html> 查阅 Bash 参考手册。

3.9.4 移动与重命名文件

在 Linux 上移动文件相当简单。使用 mv 命令可以把一个文件或目录从一个地方移动到另一个地方。为了移动一个文件,你必须拥有该文件的写权限和目标位置的写权限。

列表 3-17 演示了如何移动文件。

列表 3-17 移动文件

```
$ mv -i ~/myfile /home/bjones/yourfile
```


列表 3-17 中的命令把一个叫作 `myfile` 的文件从主目录移动到 `/home/bjones`，并把它重新命名为 `yourfile`。`-i` 选项再次确保在目标文件已经存在时给予提示。使用 `mv` 命令还可以给文件重命名。

```
$ mv -i ~/myfile ~/mynewfile
```

对目录可以做同样的事情。

3.9.5 删除文件

删除文件使用 `rm` (`remove`) 命令。对于任何主机，删除文件都要慎重。不过，在 Linux 上与 Windows 上不同，没有一个快速而简单的恢复删除文件的方法，因此要小心，并在删除文件之前做一些预防措施。第一个预防措施是在 `rm` 命令里使用 `-i` 开关。`-i` 开关打开交互模式，每次一个文件被删除时它都会给予提示。你必须回应 `y` 或 `Y` 删除文件，或者按其他键退出操作，如列表 3-18 所示。

列表 3-18 `rm` 中的 `-i` 开关

```
$ rm -i /home/jsmith/myfile
rm: remove regular file '/home/jsmith/myfile'? n
```

■提示：许多发行版把 `rm -i` 的别名命名为 `rm`，以便强制进行删除检查。输入命令 `alias` 可以查看当前主机上所有的别名列表。你也可以创建自己的别名。至于使用方法，你可到 `man` 页查看 `alias` 命令。

你还可以使用 `-r` 开关删除目录及其内容，如下所示。

```
$ rm -r /home/jsmith/backup
```

这将删除 `/home/jsmith/backup` 目录及其所有内容。

你还可以使用 `-f` 或叫强制开关忽略 `-i` 开关，如下所示。

```
$ rm -fr /home/jsmith/backup
```

这也会删除备份目录及其所有内容，但是不会得到删除的确认提示。你要小心使用这个开关，在执行这个命令时一定要意识到所在目录树中的位置——对该命令不恰当的使用可能导致破坏性的结果！

■注：不像 Windows 里有回收站，在 Linux 上删除文件往往是永久性的，除非做了备份。不过，有一些恢复文件的方法，可以到 <http://recover.sourceforge.net/unix/> 了解。但是这些方法都不推荐使用。你应该总是非常小心地删除文件，而且应该确保做了适当的备份。这在编辑配置文件时特别重要。你要在编辑、移动或删除文件之前总是做备份。

3.9.6 链接文件

在 Linux 主机上还可以创建文件的链接。链接可以像 Windows 快捷方式那样使用，不过它

有两种形式——硬链接和软链接（或称符号链接）。硬链接与 Windows 快捷方式不一样。它们实际是对物理文件的引用。如果删除了某个文件的所有硬链接，那么它们所引用的文件也被删除。

■注：因此，硬链接只能在物理分区或硬盘驱动器上创建；不能链接到一个位于别的驱动器或分区的文件。

软链接或符号链接更像 Windows 里的快捷方式：如果它们被删除了，原始文件仍然存在，只有链接被移除了。

使用 `ln` 命令可以创建链接。该命令默认创建硬链接，通过添加 `-s` 开关可以创建软链接。`ln` 命令有几种使用方式，为目标文件创建链接的最简单的方式如下所示。

```
$ ln -s /home/jsmith/myfile
```

上一行命令会创建一个 `/home/jsmith/myfile` 文件的符号链接 `myfile`。你可以查阅 `ln` 命令的 `man` 页以了解 `ln` 命令的其他可用选项。

3.9.7 编辑文件

Linux 提供了各种文件编辑工具，包括 GUI 工具和命令行工具。在 GUI 里可以找到像 `kate` 或更简单的 `gedit` 这样的编辑器。这些是简单的编辑器，但不是字处理器——很像 Windows 里的 Notepad。同样，命令行里也有编辑文件的工具，比如流行的 `vim`、`nano`、`joe` 或者异常流行的 `emacs`。

我们先快速浏览一下 `vim`，它是较老的 UNIX 编辑器 `vi` 的加强版文本编辑器。要编辑一个文件，你可以运行 `vim` 命令并指定要编辑的文件名。

```
$ vim ~/newfile
```

■提示：一些发行版还为 `vim` 命令命名一个 `vi` 的别名，以便于习惯老名字的人使用。

它打开一个位于主目录的名为 `newfile` 的文件。要往文件插入文本，键入 `i`，就是 `insert` 的简写。你就会看到 `-- INSERT --` 这个词出现在屏幕的底端。这意味着当前处于插入模式，允许往文件添加内容。现在就可以在文件里打字了。让我们键入 `hello jsmith`。还可以使用箭头键在行间和整个文件中移动。使用 `Enter` 键可以往文件添加行。

■提示：你可以使用 `touch` 命令创建空文件。直接输入 `touch` 和想要创建的文件名，例如 `touch /home/jsmith/newfile`。

当做完这些后，按 `Esc` 键。`Esc` 键使你离开插入模式（可以看到文字 `-- INSERT --` 从屏幕底端消失了）。现在可以输入冒号和字母 `w`、`q`，即 `:wq`，以保存添加到文件中的内容。它的意思是写入并退出。你还可以只指明 `w`，即只写而不退出。如果退回到命令行，你就可以查看文件，可以看到刚键入的文本。

```
$ cat newfile
hello jsmith
```


■提示：在http://blog.interlinked.org/tutorials/vim_tutorial.html上可以找到一个对 vim 的介绍，或者在命令行上运行 vimtutor 启动一个 vim 教程。

还有多种 GUI 编辑器可以使用。有些较简单，风格与 Microsoft Windows 的 Notepad 或 WordPad 应用软件相似，其他的则是比较完整复杂的字处理器和文本编辑器。

Gnome 默认自带的文本编辑器 gedit 就是其中一个例子。在 Gnome 中，通过单击“应用程序”菜单，打开“附件”标签，选择“文本编辑器”应用程序可以启动 gedit。gedit 编辑器的界面如图 3-9 所示。

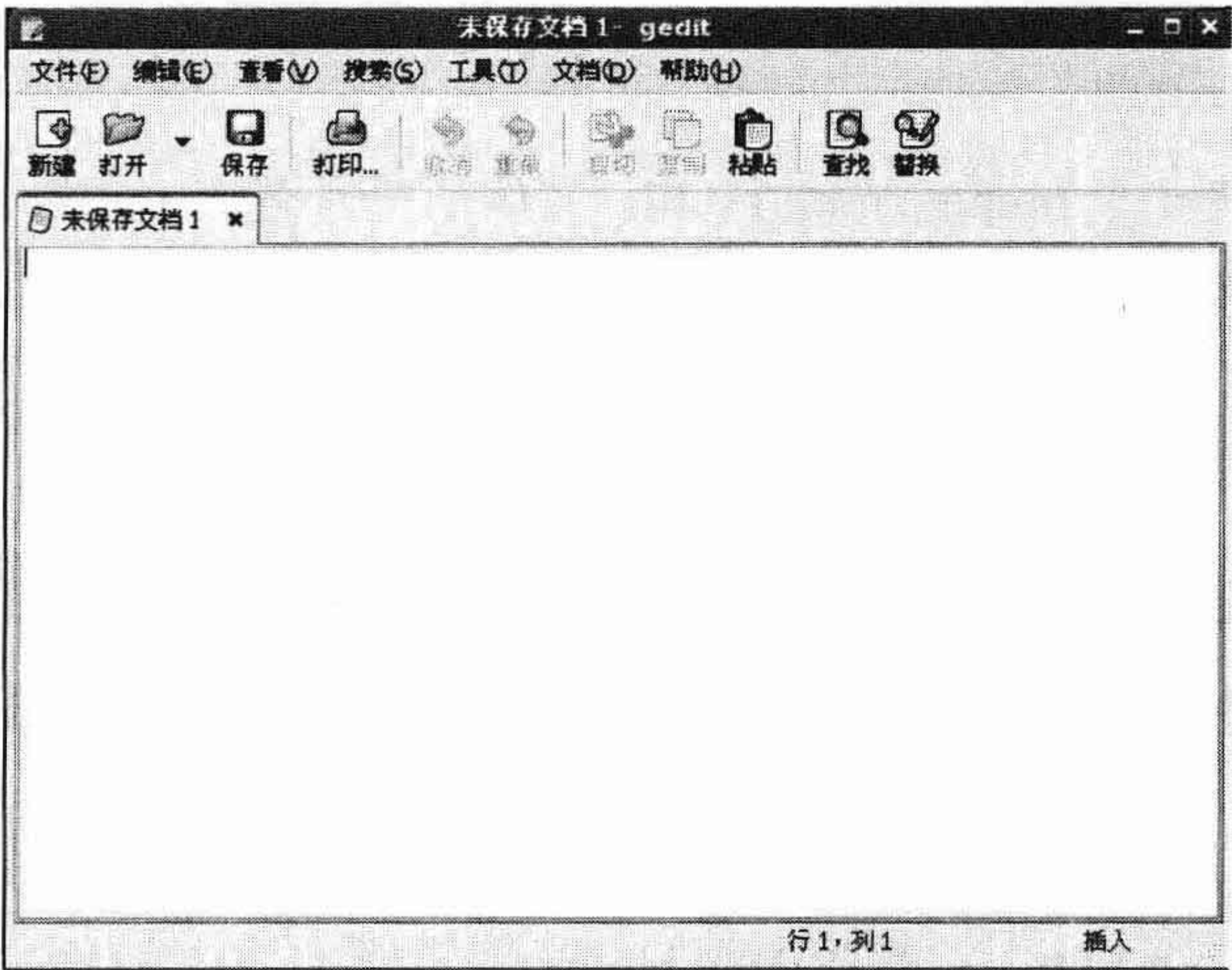


图 3-9 gedit 编辑器

另一个例子是 Kate (<http://kate-editor.org/>)，它来自 KDE 图形化界面。图 3-10 显示的是一个开着 Kate 的 KDE 桌面。



图 3-10 Kate 编辑器

3.10 小 结

那么，你学到了什么？首先，Linux 与 Microsoft Windows 的差别并不是那么大。它们有着很多相似的概念和原则，这将有助于对 Linux 的学习。通过本章的介绍，你要掌握如下一些关键内容。

- 使用 LiveCD 可以测试很多东西，而且对现有的主机和数据没有任何风险。
- 学习使用命令行——它有用且强大（有时没有其他选择）。Bash shell 特别强大，你会发现如果没有命令行 shell，将不知道如何工作。
- 对 globbing 和正则表达式多加了解——在管理主机时，你会发现两者都是军械库中 有用且强大的工具。
- 不要使用 root 用户管理主机。可利用 sudo 替代它。下一章会详细讲述 sudo 命令。
- 使用 Linux 的 man 页来查看每条命令的详细信息。这些都是信息的有用资源。
- 学习使用 vim 编辑器（或者所选择的其他编辑器），这有助于在命令行上操作配置文件。
- 如果需要帮助，不要忘记在线资源或 Linux 社区。很多人使用 Linux，有人可能已经找到了你所遇问题的解决方案。使用 Google 或你喜欢的搜索引擎寻找这些解决方案。下一章将更多地涉及用户和组以及如何在主机上管理它们。

第4章 用户和组



第3章介绍了 Linux 基础以及用户和用户集合（被称为组）的概念。我们讲述了用户和组如何拥有文件和对象，还说明了所有权和权限如何控制对这些文件和对象的访问。用户和组还被用来启动和运行进程。

本章将详细介绍用户和组的工作方式，不过先从登录过程以及其中的控制方式讲起。我们将说明如何创建用户和组，介绍更多的密码知识，并阐述 Linux 对主机访问的控制方式。

`su` 和 `sudo` 命令将是讨论的话题之一，这些命令使得命令可以以别的用户特别是 `root` 用户的身份运行，从而避开了以 `root` 用户登入来执行管理任务。`su` 和 `sudo` 命令对安全地管理主机很重要。

4.1 登入时发生了什么事？

第3章讲过用户登入 Linux 主机的事情，并显示了输入用户名和密码的屏幕截图。本章将对登录时发生的事情做更详细的解释，并开始探究其中一些可操作的选项和安全控制方式。

那么在输入用户名和密码之后来到命令行提示符或 GUI 界面之前，实际上发生了什么事？这个过程在不同发行版之间的区别很小，通常都执行一个叫做 `login` 的程序。`login` 程序执行下面的动作。

- 检查是否存在该用户和组以及是否允许该用户登入。
- 检查是否允许用户从特定的位置登入（例如只有某些用户可以登入控制台，或者 Linux 主机附带的显示屏）。
- 检查密码的正确性，如果不正确，允许一定次数（通常为 3 次）的重试。
- 检查密码的有效性，如果过期，提示用户设置新密码。
- 设置环境变量，如用户的主目录和路径。
- 启动 `shell` 进程。
- 呈现一个命令行提示符或者 GUI 界面。

接下来的章节将讲解这些步骤，并介绍如何配置和修改其中某些步骤以适合你的环境。我们从学习如何创建、删除和管理用户和组开始。

■注：本章将演示命令行的用户管理，但是如果更喜欢用 GUI 工具管理用户和组的话，这里展示的所有内容也都可以用 GUI 的方式做到。

4.2 用户与组操作

对主机访问的管理核心是用户和组的概念。第 3 章介绍了用户和被称为组的用户集合，并且发现它非常像存在于 Microsoft Windows 平台上的用户和组。

组织主机上用户和组可以用两种方式。一种方式是给域内的每台主机添加用户和组；另一种方式是在一两个身份验证服务器上统一管理用户。本章将讲述前者，而后者将在第 16 章介绍。

与在 Microsoft Windows 主机上一样，想要登入主机的每个人都需要一个已创建用户。许多应用程序，如网络和邮件服务器，也会要求创建一个用户。当它们启动时，这些应用程序会利用该用户的权利和特权访问系统资源，如文件、网络或主机的其他方面。

Linux 主机上的每个用户还要属于至少一个组，还可以属于任意数量的其他组。组是用户的集合，它们因为相似或都需要访问某个资源而集合在一起。例如，组织中销售部门的所有用户可以属于一个叫作 sales 的组。可以配置主机以确保只有 sales 组里的用户可以访问销售部门的应用程序和文件。

■注：在安装应用程序时，它们通常附带安装那些应用程序运行所需的用户和组。

使用两个命令很容易创建用户和组：useradd（创建用户）和 groupadd（创建组）。另外，可用来修改已有用户和组的两个命令是 usermod 和 groupmod。最后，为了结束生命周期，删除用户和组可以使用 userdel 和 groupdel 命令。

■提示：在 Red Hat 和 Ubuntu 上对用户和组的处理方式十分相似，它们使用许多相同的命令和选项。我们会介绍这两个版本之间的所有细微差别。

4.2.1 sudo 介绍

在解释用户和组的创建方式之前，我们想讲解一下在第 3 章提到过的 sudo 命令。sudo 命令使用户就像以 root 用户（Linux 里和 Windows 的管理员账号对等的用户）身份登录的那样运行命令。这是一个有用的功能，原因有 3 个。

- 它增加了安全性。
- 它使得对特权命令有更大的控制。
- 它使得对谁在主机上做了什么有更好的理解。

■注：在 Ubuntu 上使用 sudo 而不是 root 用户的另一个不错的原因是，Ubuntu 在默认情况下不开启 root 用户，根本就不能以 root 用户身份登录。

本章要用到 `sudo`，因为几乎所有用来管理用户和组的命令都需要 `root` 用户的特权才能运行。例如，只有 `root` 用户可以创建另一个用户。

当运行 `sudo` 命令时，它会提示输入密码（以确认你真是你所谓的你），然后允许使用 `sudo` 命令一段时间。这个时间在 Red Hat 上是 5min，在 Ubuntu 上是 15 min。当时间到达，它会提示你重新输入密码。

■提示：第一次运行 `sudo` 命令时，它可能还显示一个警告，要求小心使用 `sudo` 命令的权力。

在 Ubuntu 上，`sudo` 命令可以使用，并且已经为安装 Ubuntu 时所创建的用户配置好了。如果以该用户身份登入，马上就可以使用 `sudo` 命令。还可以把其他用户添加到 `admin` 组，以使它们能够访问 `sudo`。`usermod` 命令（本章稍后会详细介绍它）可以用来把用户添加到该组。

```
$ sudo usermod -G admin ataylor
```

这里使用 `sudo` 和 `usermod` 命令修改了一个叫作 `ataylor` 的用户。通过指定 `-G` 选项和该用户要添加到的组，就把它添加到了 `admin` 组。（注意，已经使用 `sudo` 命令对用户做了修改。允许做这件事的唯一用户就是安装主机时所创建的那个用户，因此必须以那个用户身份登入才能做这个改动。）

在 Red Hat 上，`sudo` 命令默认不开启，因此需要开启它。为了做到这一点，应使用一个叫做 `visudo` 的命令编辑 `sudo` 命令的配置文件 `/etc/sudoers`。为此，以 `root` 用户身份登录并运行 `visudo` 命令。

```
# visudo
```

从 `#` 命令提示符可以看出，你现在正以 `root` 用户的身份登录并正在执行 `visudo` 命令。它打开一个看起来很像 `vi` 或 `vim` 编辑器的编辑程序。这个文件的内容如下面一行所示。

```
# %wheel ALL=(ALL)    ALL
```

你需要把这行的注释符号去掉，并使用和 `vim` 里一样的命令写文件然后退出，即键入冒号、`w` 和 `q`（`:wq`），然后回车。于是一个叫作 `wheel` 的组中的所有成员都可以使用 `sudo` 命令。然后你可以把一个用户添加到 `wheel` 组，如下所示。

```
# usermod -G wheel ataylor
```

这里再次指定组 `wheel`，最后是 `-G` 选项和希望添加到该组的用户名。现在 `ataylor` 就可以使用 `sudo` 命令了。

4.2.2 创建用户

既然理解了如何打开和使用 `sudo` 命令，现在就可以开始看看用户和组了。让我们从使用 `useradd` 命令创建新用户开始，如列表 4-1 所示。

列表 4-1 创建新用户

```
$ sudo useradd -m -c 'John Smith' jsmith
```

■注：如列表 4-1 所示，以 `sudo` 命令作为 `useradd` 命令的前缀避免了以 `root` 用户登录的必要性。

`useradd` 命令有许多选项，列表 4-1 中只使用了其中的两个。第一个选项 `-m` 告诉主机为该用户创建一个主目录。主目录名字和位置的格式通常类似于 `/home/username`。

提示：可以预先组建好新的主目录，比如使用类属配置文件组建。要做到这一点，你需要往 `/etc/skel`（`skeleton` 的简写）目录添加文件。当创建一个新的主目录（使用 `-m` 选项）时，新目录下的所有文件都会被复制到用户的新主目录里。

`-c` 选项添加对新用户的说明。这个说明保存在 `/etc/passwd` 文件里。所有用户在这个文件里都有一个输入项，本章稍后会分析这个文件以及用来存储组数据的 `/etc/group` 文件。最后，指定新用户的名字 `jsmith`。

默认情况下，新用户创建时不能使用而且没有设置密码。需要用 `passwd` 命令（本章稍后将更详细地介绍它）修改他的密码。

表 4-1 列出了 `useradd` 其他一些有用的命令行选项。

表 4-1 useradd 的一些命令行选项	
选项	说明
<code>-c</code>	添加对用户的说明
<code>-d homedir</code>	用户的主目录
<code>-m</code>	创建用户的主目录
<code>-M</code>	不创建用户的主目录（仅用于 Red Hat）
<code>-s shell</code>	指定该用户使用的 shell

`-d` 选项允许指定用户的主目录。`-M` 选项指出 Red Hat 衍生的发行版不创建主目录。这个选项体现了在 Red Hat 和 Ubuntu 发行版上创建用户的主要差别。在 Red Hat 衍生的发行版上，主目录会自动创建。

Ubuntu 需要执行带 `-m` 选项的 `useradd` 命令，否则就不创建主目录。

注：在 Ubuntu 上创建用户的一个备选方法见补充知识“`ADDUSER`：Ubuntu 上的一种备选方法”。

最后，`-s` 选项允许为该用户指定一个非默认 shell。

提示：推荐阅读 `useradd` 命令的 man 页，以获得关于此命令的更详细信息。

用户默认设置

新用户创建时带许多默认值（如用户的 shell 设置）。那么 `useradd` 命令从哪里获得这些默认值呢？在 Red Hat 和 Ubuntu 两个发行版上，默认设置都存储在 `/etc/default/useradd` 文件里，你可以使用下面的命令显示当前的默认设置。

```
$ sudo /usr/sbin/useradd -D
```

列表 4-2 显示了该文件的一个例子。

列表 4-2 /etc/default/useradd 文件

```
$ sudo cat /etc/default/useradd
# useradd defaults file
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
```

该文件通常在安装主机时默认组建，不过可以修改它以适应具体环境。表 4-2 显示了可以包含在 useradd 文件里的可能选项。

表 4-2 /etc/default/useradd 文件	
选项	说明
SHELL	默认 shell 的路径
HOME	用户的主目录路径
SKEL	用来提供用户新主目录里默认内容的目录
GROUP	默认的组 ID
INACTIVE	密码过期后可以修改的最大天数
EXPIRE	用户账号的默认到期日期

该文件中的每个选项都控制一个特定的默认设置；例如，SHELL 选项为用户指定默认 shell。HOME 选项指定所有新用户被创建时所在的目录。SKEL 选项指定用哪个目录构建用户的主目录，如前所述，它的默认值是/etc/skel。GROUP 选项指定要使用的默认组 ID (GID)，通常不要去改它。接下来的章节里会较详细介绍组、成员和 GID。

最后，两个不同的选项 INACTIVE 和 EXPIRE 管理两种不同类型的用户账号期限。INACTIVE 值控制用户的密码过期后多少天用户可以重置他的密码。它允许指定一个天数，如果某个用户的密码过期了，在该用户被标记为无效之前的限定时间内，他可以重置密码。-1 的设置会关闭该设置，而 0 的设置使密码一过期该账户就失效。

■注：本章稍后将介绍密码期限。

EXPIRE 选项在创建临时账号时有用，它指定一个 YYYY-MM-DD 格式的日期，在这个日期该账号会到期并失效。EXPIRE 这个默认设置允许为所有账户指定一个这样的日期。你还可以在命令行使用下面的命令创建一个独立的账号。

```
$ sudo useradd -e 2009-09-15 temp_account
```

该命令创建了一个叫 temp_account 的账号，它将在 2009-09-15 这一天失效。

通过执行带-D 选项的 useradd 命令可以改变这个文件里的许多默认设置。列表 4-3 显示了如何为新用户更改默认 shell，表 4-3 显示了可与-D 选项一起使用的附加选项。

列表 4-3 用带-D 选项的 useradd 命令更改默认设置

```
$ sudo useradd -D -s /bin/bash
```


■提示: 你还可以用 `chsh` 命令更改默认 shell。在 Red Hat 上使用 `chsh -l` 可以看到所有可用 shell 的列表。在 Ubuntu 上可以在 `/etc/shells` 文件里看到这个列表。

表 4-3 useradd -D 的选项	
选项	说明
-b path/to/default/home	指定新用户主目录的路径前缀
-e date	指定默认的到期日期
-f days	指定密码到期后账号失效之前的天数
-g group	指定默认的组
-s shell	指定默认的 shell

4.2.3 创建组

如前所述，每个用户必须属于至少一个组。默认情况下，在大多数 Linux 发行版上，包括 Red Hat 和 Ubuntu，在创建一个用户的同时还创建与该用户同名的新组。新用户永远是这个组的唯一成员。

■注: 为每个用户创建唯一组的做法被称为用户私有组（user private group, UPG）模式。这是一个管理组权限的灵活模型。关于 UPG 的资料可以到 http://www.centos.org/docs/5/html/5.1/Deployment_Guide/s1-users-groups-private-groups.html 查找。

在我们的例子中，第一个用户 `jsmith` 将自动属于叫作 `jsmith` 的组。这个组被称为基本组。该用户也可以属于其他组，这些另外的组被称为附加组。

那么如何知道新用户属于什么组？为了查看一个特定用户的资料，你可以使用 `id` 命令，如列表 4-4 所示。

列表 4-4 id 命令

```
$ id jsmith
uid=1003(jsmith) gid=1003(jsmith) groups=1003(jsmith)
```

列表 4-4 使用 `id` 命令查询新用户 `jsmith`。但是这个命令返回一些关于 `uid` 和 `gid`、用户名和一些数字的神秘信息。那么这些信息到底是什么意思？

每个用户和组在创建时被分配一个唯一的用户 ID（UID）和组 ID（GID）。UID 的范围从 0 到 65535，`root` 用户的 UID 总是 0。GID 的范围也是从 0 到 65535，`root` 用户的 GID 也总是 0。

如果对 `root` 用户运行 `id` 命令，可以看到下面的结果。

```
$ id root
uid=0(root) gid=0(root) groups=0(root)
```

它显示 `root` 用户有一个编号为 0 的 UID 和一个编号为 0 的 GID。

■注: 主机上每个用户和组必须有一个唯一的 UID 和 GID。不能在主机上使用与已有用户或组相同的 UID 或 GID 创建用户或组。操作系统会自动分配这些号码，并防止任何冲突发生。

大部分发行版为特殊类型的用户和组预留一些号码范围。例如，Red Hat 发行版为运行服务的“系统”用户和组——如运行数据库或网络服务器的用户——预留了 1 到 499 的 UID 和 GID 范围。Ubuntu 为同样的用途预留了 1 到 999 的 UID 和 GID 范围。因此，Red Hat 主机上创建的第一个新用户的 UID 是 500，GID 也是 500。而 Ubuntu 上第一个新用户的 UID 和 GID 都是 1000。

提示：你可以控制/etc/login.defs 文件提供给用户的 UID 和 GID 范围。对于 UID，可编辑 UID_MIN 和 UID_MAX 修改其范围。对于 GID，可编辑 GID_MIN 和 GID_MAX 调整其范围。你可能永远不会想修改它，但这个选项是存在的。

因此，在列表 4-4 中对 jsmith 用户执行了 id 命令，它显示用户的 UID 和 GID 都是 1003（用户名和组名在 UID 和 GID 后的括号里）。最后一个字段 groups 是显示基本组和附加组的地方。

把用户添加到组有两种方法。第一，可以在创建用户时用 useradd 命令添加到组。第二，可以使用 usermod 命令修改一个已有用户，把他添加到组。

下面一行命令要创建第二个用户，叫作 ataylor，并在创建时把她添加到一些组。

```
$ sudo useradd -m -c 'Anne Taylor' -G printing,finance ataylor
```

这里指定的-G 选项允许把新用户 ataylor 想要加入的组以一个由逗号隔开的组清单的形式给出。-G 选项允许用户加入另外的组而不是她的基本组（即创建该用户的同时创建的与用户同名的那个唯一组）。在这里用户 ataylor 是一个叫作 ataylor 的唯一 UPG 基本组的一名成员，我们正试图把她加入另外的附加组 printing 和 finance。

如果马上执行这个命令，一定会失败，因为所有这些组都要在添加用户之前存在，否则将出现一个错误消息并且该用户创建失败。在 Ubuntu 上，这种情形下产生的错误消息如下所示。

```
useradd: unknown group printing
useradd: unknown group finance
```

在 Red Hat 上，错误信息稍微有点不同。

```
useradd: invalid numeric argument 'printing'
```

那么在这种情况下，需要先创建组，可以通过 groupadd 命令来执行，如列表 4-5 所示。

列表 4-5 创建新组

```
$ sudo groupadd printing
$ sudo groupadd finance
```

表 4-4 显示的是 groupadd 命令的一些可用命令行选项

表 4-4 groupadd 的命令行选项	
选项	说明
-g GID	为组设置 GID。它必须是唯一的号码
-r	创建一个系统组（GID 在系统 GID 范围内）

如果希望覆盖自动生成的 GID 号，请使用 `-g` 选项和一个具体的号码。`-r` 选项只在 Red Hat 上可用，你可以创建一个系统组并确保该组被分配一个系统组范围内的 GID（例如，在 Red Hat 主机上为 1 到 499）。

在试图创建 `ataylor` 用户时，我们成功了，因为作为先决条件的组现已存在。

```
$ sudo useradd -m -c 'Anne Taylor' -G printing,finance ataylor
```

我们还可以使用 `usermod` 命令把已有的用户添加到组。

```
$ sudo usermod -a -G accounts ataylor
```

`usermod` 命令用来修改已有的用户。通过指定 `-a`（追加的意思）选项、`-G` 选项和要加入的新组名（这个组必须已经存在），我们把 `ataylor` 用户添加到了 `accounts` 组。

■ **提示：**你可以使用 `usermod` 命令更改一个用户的许多参数。推荐阅读它的 `man` 页以获得进一步的信息。

`gpasswd` 命令也可以用来管理组，它允许把管理组及其成员的职责委托出去。你可以把向某个组添加或删除用户的权利指派给某个用户。例如，你可以让销售团队里的某个人管理 `sales` 组的成员。关于 `gpasswd` 的更详细信息可以到 `man` 页了解。

4.2.4 删除用户和组

除了创建和修改用户与组，还希望能够删除它们。可以通过下面两个命令实现删除任务：`userdel` 和 `groupdel`。`userdel` 命令删除用户，`groupdel` 命令移除组。现在让我们使用 `userdel` 命令删除早前创建的 `ataylor` 用户，如列表 4-6 所示。

列表 4-6 删除用户

```
$ sudo userdel ataylor
```

`userdel` 命令删除了该用户，但是默认情况下不会删除用户的主目录。可以使用 `userdel` 命令的 `-r` 选项强制 Linux 删除该用户的主目录。它会删除 `/home/username` 目录和它里面的所有文件，但不会删除这个目录之外可能也属于该用户的文件。`userdel` 命令也不会删除当前已经登录主机的用户。

对拥有文件的用户执行删除操作可能有问题。如果删除了一个用户，那么该用户的所有对象将不再归他所有。这些对象可以识别，因为文件列表中的用户名会替换为以前的 UID（对于已删除的组也是这样）。那么，如果再创建用户时使用这个 UID 或 GID，新用户就会立刻拥有已删除用户的文件。一个非常好的主意就是，在删除用户之前确认该用户拥有的文件和目录并制定针对它们的处理方案。本章稍后会说明如何指定文件和目录的所有权。考虑到这个问题，停用一个用户有时比删除它更好。但是如果一定要删除一个用户，你可以运行命令 `find / -user UID -o -group GID` 来查找与已删除用户相关联的全部文件。

要删除一个组，我们使用 `groupdel` 命令并指定要删除的组的名字。

```
$ sudo groupdel finance
```


这个命令会从主机移除这个组。重点要指出的是，`groupdel` 命令不能删除用户的基本组——比如说，在删除 `ataylor` 用户之前不能删除 `ataylor` 组。如果想删除用户的基本组，就必须先删除该用户。与删除用户时的情况一样，删除组会让那些组所拥有的文件孤立。

补充知识：ADDUSER：Ubuntu 上的一种备选方法

Ubuntu 装载了两个附加的用户管理实用工具 `adduser` 和 `addgroup`。它们易用又方便，是 `useradd` 和 `groupadd` 命令的替代品。`adduser` 一般要带被创建的新用户的用户名运行。然后这个工具会要求提供别的信息。举个例子，让我们为用户 Anne Taylor 添加一个账号。

```
$ sudo adduser ataylor
Adding user `ataylor' ...
Adding new group `ataylor' (1001) ...
Adding new user `ataylor' (1001) with group `ataylor' ...
Creating home directory `/home/ataylor' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for ataylor
Enter the new value, or press ENTER for the default
  Full Name [ ]: Anne Taylor
  Room Number [ ]:
  Work Phone [ ]:
  Home Phone [ ]:
  Other [ ]:
Is the information correct? [Y/n] y
```

`adduser` 命令寻找它所需的全部变量，然后调用带正确参数的 `useradd` 命令创建账号。这意味着即使在使用 `adduser` 命令时，`/etc/default/useradd` 里所配置的默认 `useradd` 选项也仍然被遵守。

你还可以使用 `adduser` 脚本把用户快速添加到组，就是运行下面这个命令。

```
$ sudo adduser username groupname
```

这里的用户和组必须已经存在于主机上。

`adduser` 和 `addgroup` 脚本本身也可以通过 `/etc/adduser.conf` 文件配置。默认情况下，它们会按照指定的名字创建用户，把它们放到一个同名的组里，创建主目录，并分配可用范围内最低的用户 ID 和组 ID 号。

4.2.5 通过 GUI 管理用户和组

Red Hat 和 Ubuntu 两者都有管理用户和组的图形化用户界面（graphical user interfaces, GUI）。Red Hat 上的 GUI 工具被称为“用户管理者”，可通过选择“系统”→“管理”→“用户与组群”启动（见图 4-1）。

在这个非常简单的界面里可以添加、更改和删除用户和组，如图 4-2 所示。

单击“添加用户”按钮可以添加用户，单击“添加组群”按钮可以添加组。主机当前的用户和组的清单包含在两个标签里。你可以选择一个用户或组，然后单击“属性”按钮编辑用户和组。在这个界面你可以配置用户和组的各个方面。要删除一个用户或组，选定它然后

单击“删除”按钮即可。

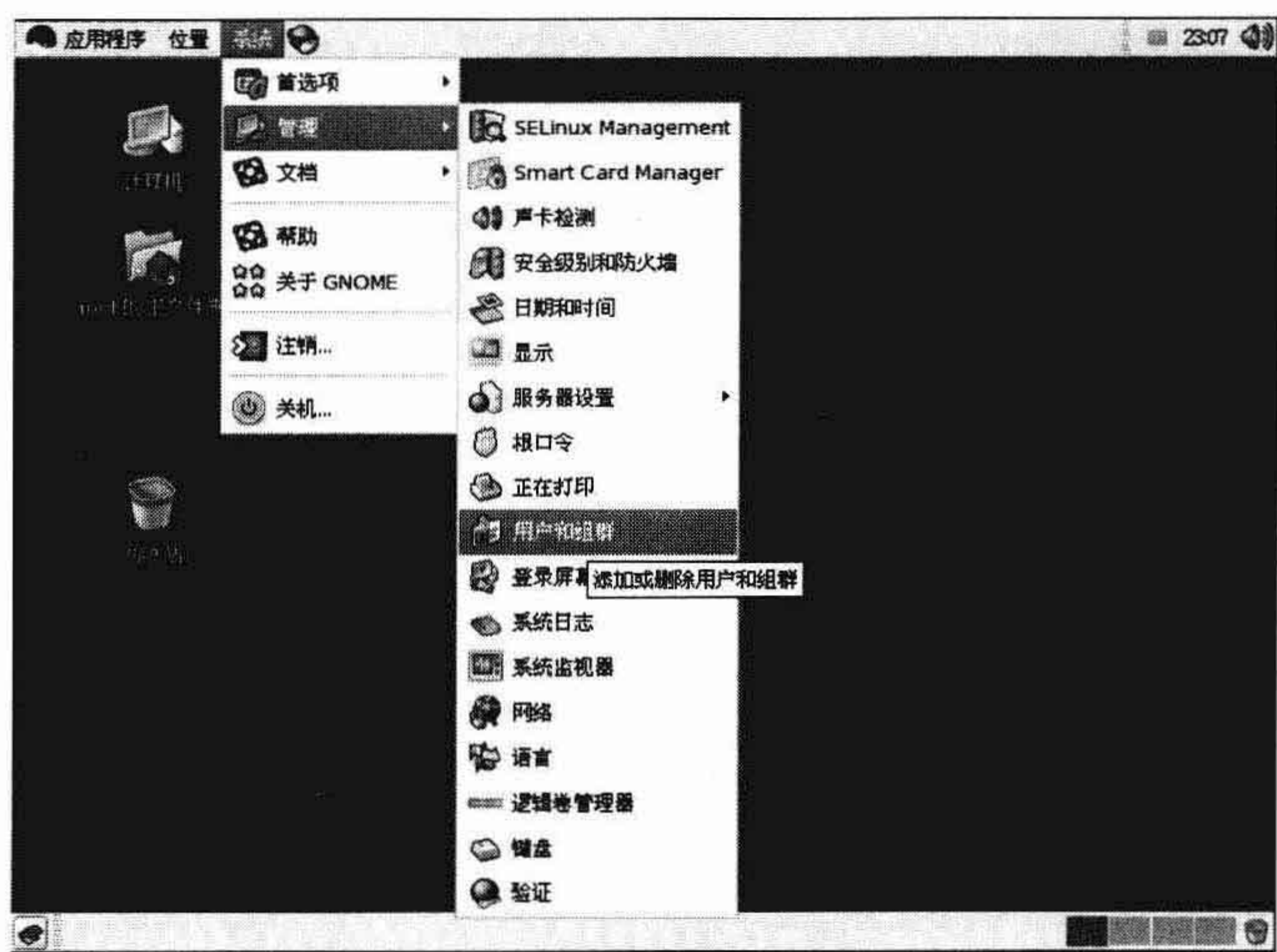


图 4-1 打开“用户管理者”

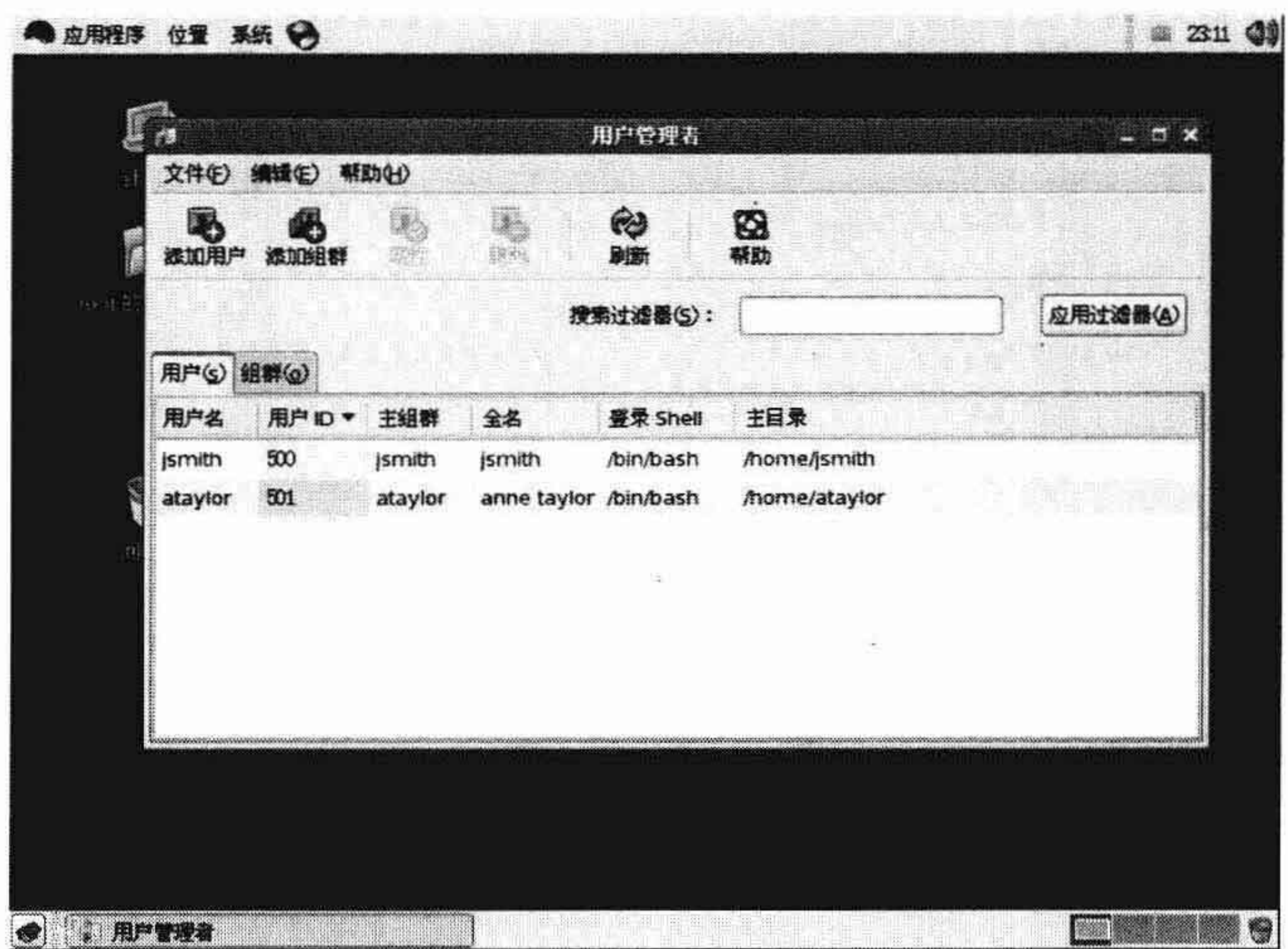


图 4-2 “用户管理者”的界面

在 Ubuntu 上，GUI 用户管理工具被称为 Users Settings。可以通过 System→Administration→Users and Groups 启动它，如图 4-3 所示。

Users Settings 是一个简单的用户管理器，它显示当前主机上的全部用户（见图 4-4）。

要管理用户，单击“Unlock”按钮，它发起与 sudo 命令等效的图形化任务，并提示输入密码。如果输入密码成功，你就可以管理用户和组了。通过单击“Add User”可以添加一个用户，通过选中用户并单击“Delete”可以删除一个用户。通过选中用户并单击“Properties”按钮可以编辑用户的设置。要管理组，单击“Manage Groups”按钮。

■注：记住，Ubuntu 服务器版本默认不带 GUI，你要利用所提供的命令行工具。Ubuntu 桌面发行版带相应的 GUI 工具，因为它默认安装一个 GUI。

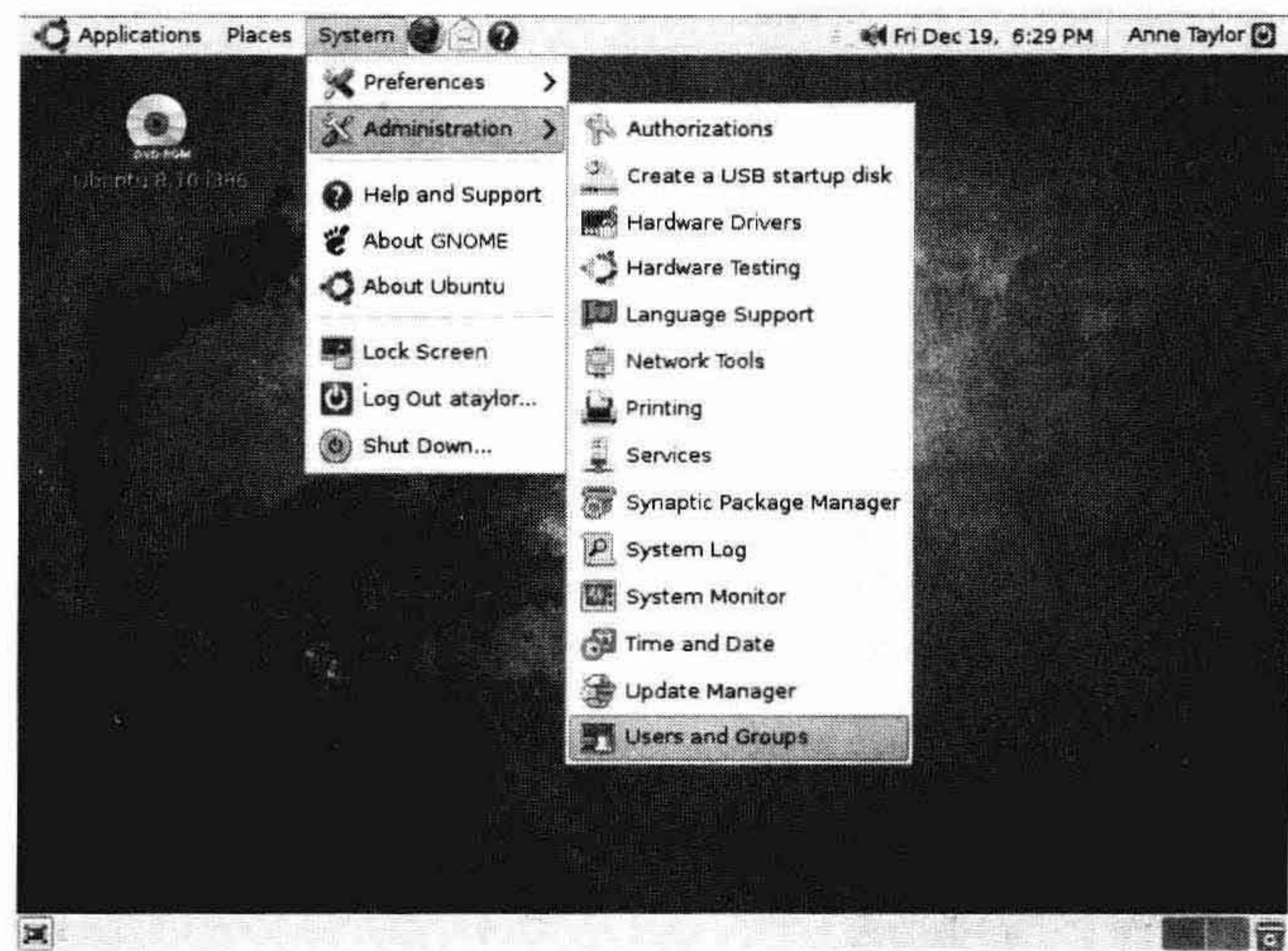


图 4-3 打开 Users Settings

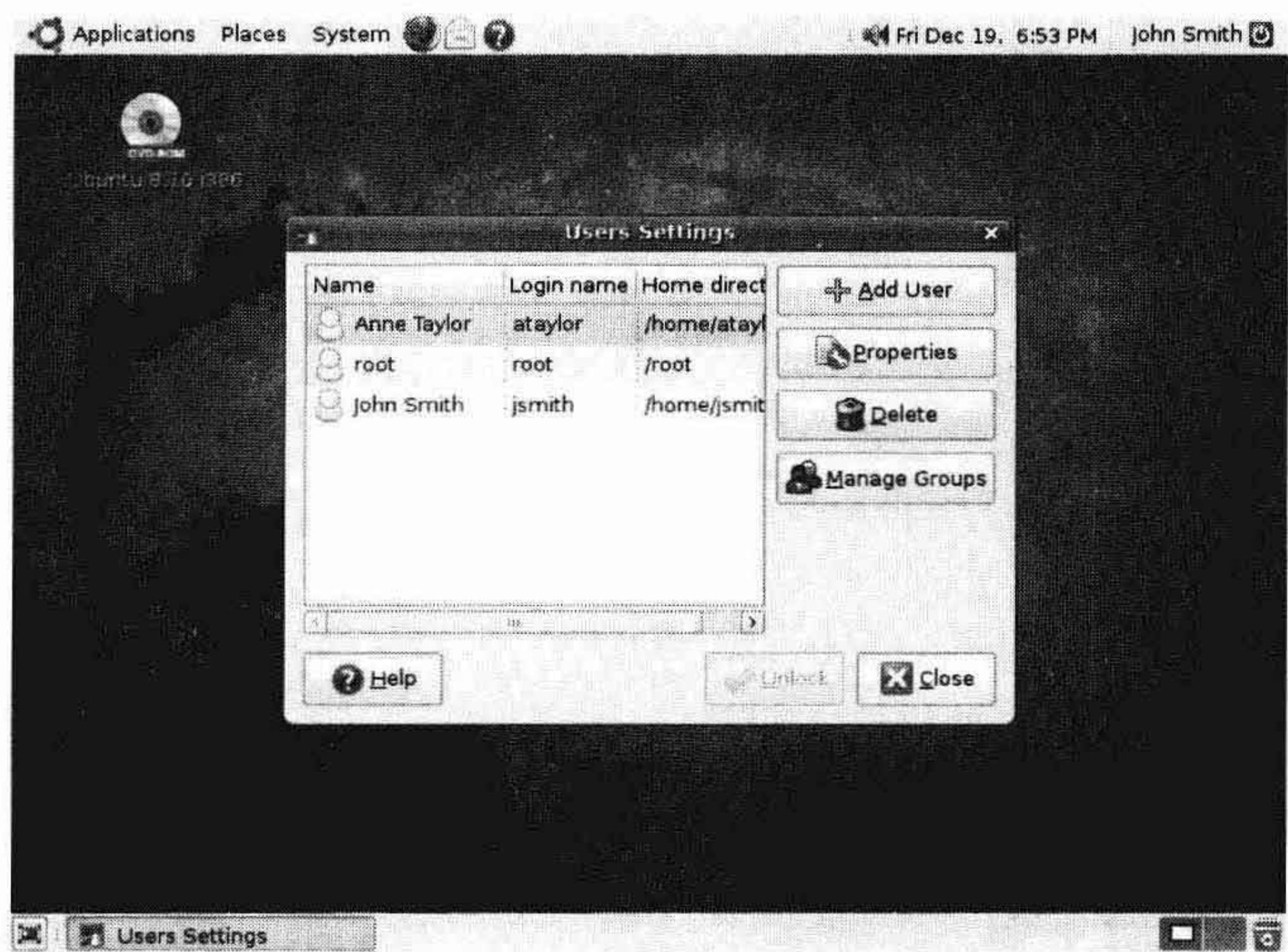


图 4-4 Users Settings 界面

4.2.6 密码

你既然创建了一个新用户，那么就可能想设置或修改该用户的密码。要做这些事，请使用 `passwd` 命令。`passwd` 命令有两种工作方式，采用哪种方式取决于谁在运行该命令。如果像 `ataylor` 这样的普通用户运行该命令，她会得到更改她自己密码的提示。在列表 4-7 中可以看到运行中的 `passwd` 命令。

列表 4-7 更改密码

```
$ passwd
Changing password for ataylor.
(current) UNIX password:
```


Enter new UNIX password:
Retype new UNIX password:

键入当前的密码，然后键入新密码。它提示输入两次新密码以确保它的正确性。同时，所提供的密码必须是合适的。大多数发行版一般都会执行一些基本的密码检查，以尽量防止所提供的密码太弱或太容易猜。在大部分发行版上一般都有如下这些检查要求。

- 密码长度最少 4 个字符。
- 不是回文（即上次密码的倒置）。
- 不能和上次密码相同而只改变字母的大小写（即 password 变为 PASSWORD）。
- 基本的相似性检查。
- 基于密码长度和字符组合的复杂性测试（全字母、全数字，等等）。
- 简单的轮转检查（即用一个密码进行字母轮转，比如 ginger 改为 ingerg）。

如果所提供的密码不够复杂，它会给出一个错误消息表示密码有问题。然后它会提示提供一个比较可接受的密码。

另外一种情况就是，如果以 root 用户身份运行 passwd 命令，那么就可以如列表 4-8 那样更改其他用户的密码。

列表 4-8 更改别人的密码

```
$ sudo passwd jsmith
```

在列表 4-8 中，passwd 命令提示为用户 jsmith 提供一个新密码。

■提示：值得注意的是，作为 root 用户，任何有关密码不好和更改的用户密码太弱或太容易猜的警告都可以不管。

4.2.7 密码时效

密码时效的概念允许为密码指定一个有效的时间期限。时间期限一到，就要求用户选择一个新密码。这样做的好处是确保密码定期更改，即使一个密码被盗、被破解或者为前雇员所知，它的价值也是有时限的。不幸的是，对许多用户来说，定期修改密码的需要会增加他们把密码写下来的欲望。我们推荐根据主机的性质对大多数密码选择使用期限为 30 到 60 天之间。更重要的主机应该有更短的密码期限（如 30 天），而较不重要的主机可以有较长的期限。一些组织选择同一个使用期限，那么对所有主机上的所有用户的密码期限都是一致的。

密码使用期限的处理方式有两种。第一种方式使用被称为 chage 的命令行工具单独更改一个用户账号的密码期限。列表 4-9 显示了这个命令。

列表 4-9 chage 命令

```
$ sudo chage -M 30 ataylor
```

列表 4-9 使用-M 选项把用户 ataylor 的密码使用期限设置为 30 天。30 天后该用户的密码到期，然后会得到输入新密码的提示。表 4-5 显示了几个可设置的其他变量。

表 4-5 chage 命令的标志

选项	说明
-m days	设置两次密码改动之间的最小天数。0 值允许用户随时更改密码
-M days	设置一个密码保持有效的最大天数
-E date	设置一个用户账号到期的日期，该用户账号在这个日期自动停用
-W days	设置密码过期之前用户被警告修改它的天数
-d days	设置密码上次修改的天数（从 1970 年 1 月 1 日算）
-I days	设置密码过期之后账号被锁定的天数

■提示：在 Unix/Linux 领域经常会遇到 1970 年 1 月 1 日这个日期。这个日期还以 Unix 纪元或 Unix 时间为大家所知。它被用来描述时间点，测算从 1970 年 1 月 1 日开始的秒数（例如，1229519557）。可以使用命令 `date +%s` 找到主机上的 Unix 时间。

第一个选项 `-m` 允许指定两次密码改动之间的最小时间。如果设置为 0 值，用户就可以随时更改密码。选项 `-W` 规定用户密码过期之前多少天他会收到一个密码即将到期的警告。`-d` 选项主要对直接终止一个密码有用。通过把 `-d` 选项设置为 0，该用户的上次密码改动日期就变为 1970 年 1 月 1 日，如果 `-M` 选项大于 0，那么该用户必须在下次登录时更改他的密码。最后一个选项 `-I` 提供了一个以天为单位的期限，那个期限之后过期未更改密码的用户账号会被锁定，并因此不能用来登录。

如果不带任何选项地运行 `chage` 并且只指定用户，那么它会启动一连串交互式的提示来设置必需的值，如列表 4-10 所示。方括号 `[]` 中的值表示所设置的用户密码时效的当前值。

列表 4-10 运行不带选项的 chage

```
$ sudo chage ataylor
Changing the aging information for ataylor
Enter the new value, or press return for the default
Minimum Password Age [0]:
Maximum Password Age [30]:
Last Password Change (YYYY-MM-DD) [2009-06-27]:
Password Expiration Warning [7]:
Password Inactive [-1]:
Account Expiration Date (YYYY-MM-DD) [2009-07-28]:
```

用户还可以利用带 `-l` 选项的 `chage` 命令显示密码应当到期的时间。

```
$ chage -l ataylor
```

密码期限的另一种处理方法是在 `/etc/login.defs` 文件里为所有用户进行默认设置。

列表 4-11 显示了 `/etc/login.defs` 里可控制密码期限的选项。

列表 4-11 login.defs 里密码期限的控制选项

```
PASS_MAX_DAYS 60
PASS_MIN_DAYS 0
PASS_WARN_AGE 7
```

在列表 4-11 中，通过使用 `PASS_MAX_DAYS` 选项把最大密码期限设置为 60 天。通过把

PASS_MIN_DAYS 选项设置为 0，允许用户在任何时间更改密码，并且使用 PASS_WARN_AGE 选项，在用户的密码到期前 7 天向他们发出一个密码将到期的警告。

4.2.8 停用用户

作为 root 用户，你还可以使用 passwd 命令行和 -l 或叫锁定选项停用和启用用户账号。例如，考虑下面的命令。

```
$ sudo passwd -l ataylor
```

上述命令会锁定 ataylor 用户并阻止 ataylor 使用她的密码登入主机。然后你可以使用 -u 或叫解锁选项为用户解锁。

```
$ sudo passwd -u ataylor
```

不过，这并没有完全禁止对主机的访问。用户可以通过其他身份验证机制访问主机，如通过公钥机制使用 SSH 远程访问主机。

另外一种完全禁止用户访问的方式是使用带 --expiredate 选项的 usermod 命令。

```
$ sudo usermod --expiredate 1
```

它把该账号的到期日期设置为 1970 年 1 月 1 日，从而立即停掉了该账号。该用户现在不能在这个主机上做任何事。

最后，可以在 Ubuntu 上把登录 shell 设置为 /bin/false，或者在 Red Hat 上设置为 /sbin/nologin。这并没有锁定用户，只是使该用户不能访问 shell。

```
$ sudo usermod -s /bin/false
```

■注：还可以把用户的 shell 设置为一个命令。例如，把用户的 shell 设置为 /bin/mail 命令，它是一个小型的命令行邮件阅读器。然后当该用户登录时，她只能访问那个命令。

4.2.9 存储用户和组数据

首先，主机验证用户存在并允许登入。Linux 发行版把用户、组和其他信息的资料存储于主机的 3 个文件：/etc/passwd、/etc/shadow 和 /etc/group。一般很少需要编辑这些文件，因为有命令和工具可以添加、删除和管理用户和组。不过，了解它们包含什么信息是有用的。

■提示：如果使用其他形式的身份验证机制，如 NIS、LDAP 或 Active Directory（第 16 章将介绍这些内容），那么主机通常会查询其中一个身份验证机构的存储以确认用户存在并允许登入。

第一个文件 /etc/passwd 包含一个所有用户及其资料的清单。列表 4-12 显示了一些 passwd 输入项的例子。

列表 4-12 /etc/passwd 的输入项

```
root:x:0:0:root:/root:/bin/bash
daemon:x:2:2:daemon:/sbin:/sbin/nologin
```


每个输入项可以分解为由冒号隔开的几部分。

```
username:password:UID:GID:GECOS:Home Directory:Shell
```

用户名达到 8 个字符长度，并对大小写敏感（尽管它通常都是小写）。接下来字段里的 `x` 是密码的标记。真实的密码存储在 `/etc/shadow` 文件里，这个文件很快就会在补充知识“隐蔽密码”里介绍。

接下来是 `UID` 和 `GID`。如前所述，在 Linux 主机上，每个用户账号和组被分配一个数字 ID；用户被分配一个 `UID`，组被分配一个 `GID`。根据发行版的不同，较低的 `UID` 和 `GID` 号表示 `root` 或 `daemon` 之类的系统账号和组。在 Red Hat 上，系统账号的 `UID` 和 `GID` 是那些低于 500 的 ID，而在 Ubuntu 上那些 ID 低于 1000。

■注：如前所述，`root` 用户的 `UID` 和 `GID` 都为 0。这应该是主机上 `UID` 和 `GID` 为 0 的唯一用户。

接下来的一项是 `GECOS` 或叫注释字段（见 http://en.wikipedia.org/wiki/Gecos_field）。这个字段通常包含如用户名、办公位置和电话号码之类的数据。如果 `GECOS` 字段的数据项超过一个，就用逗号隔开每个数据项。

然后是用户的主目录。正如第 3 章所述，它通常位于 `/home` 目录（例如 `/home/jsmith`）。

最后一项是用户的默认 shell。第 3 章讲过 shell 是一个命令行平台，用户可以通过它与主机交互。每个 shell 都通过运行一个二进制文件启动。例如，为了启动 Bash shell，你要执行 `/bin/bash` 二进制代码。`/etc/passwd` 文件里指定了这个二进制文件。如果默认 shell 指向一个不存在的文件，那么该用户将不能登录。

列表 4-12 中的第 2 行使用了一个假 shell：`/sbin/nologin`，这样它不仅阻止用户登录，还把登录尝试记录到 `syslog` 后台程序。

■注：`syslog` 后台程序是 Linux 日志服务器。它接收来自操作系统和应用程序的日志输入项并把它们写入文件，这些文件一般位于 `/var/log` 目录里。第 18 章将详细介绍日志。

Red Hat 主机常用这个方法表明这个用户不能登录。在 Ubuntu 主机上这个 shell 是 `/bin/false`。大多数用户都会有一个 shell 输入项，这个输入项引用启动 shell 的二进制文件，如 `/bin/bash`。

补充知识：隐蔽密码

你可能已经注意到，在 `/etc/passwd` 里出现的不是密码而是字母 `x`。这是因为大多数（如果不是全部的话）现代发行版使用隐蔽密码进行密码管理。

以前，密码以单向散列值的形式存储在 `/etc/passwd` 里，其安全性很有限，用强力破解方法就可以知道用户名和密码。强力破解（Brute-force cracking）是一种攻击密码的方法，这种方法用数千或数百万不同的密码来尝试直至找到一个匹配的密码。`/etc/passwd` 文件特别容易受到这种攻击，因为应用程序对它的使用要求它对所有用户可读，或者说完全可读。如果从主机上偷走 `passwd` 文件的一个副本并进行离线强力破解，这就特别危险。考虑到这类密码存储于 `passwd` 文件时的弱安全性，一台现代计算机破解简单密码只是几分钟的事情，或更难一点的密码也就是几天的时间。

隐蔽密码把用户名和密码分开并把密码以散列值的形式存储在 `/etc/shadow` 文件里，有助于降低这种风险。默认情况下，所使用的是 MD5 散列值（尽管较新的发行版支持其他散列

类型，如 Blowfish)。这些 MD5 散列值较难破解，而且为了进一步保护这些密码，/etc/shadow 文件为 root 用户所有，root 是唯一可访问此文件的用户。接下来的一行在 shadow 文件里有代表性。

```
root:$1$RwETwzjv$ifht7L/HiLCPR8Zc935fd0:13675:0:99999:7:::
```

与 passwd 文件一样，你也可以把 shadow 文件分解成由冒号隔开的组件。shadow 文件的组件如下所示。

- 用户名。
- 密码。
- 上次更改密码的日期。
- 两次密码改动之间的最小天数。
- 以天为单位的密码到期时间。
- 以天为单位的密码到期警告期限。
- 密码到期到账号被禁用之间的天数。
- 账号被停用的日期。

用户名和 passwd 文件里的用户名匹配。密码本身被加密，而用户账号的状态可以通过用作密码字段前缀的两类特殊字符得知。如果密码字段以!或*为前缀，那么账号被锁定，并且不允许该用户登录。如果密码字段的前缀是!!，那么还没有设置密码，并且该用户不能登入主机。剩下的输入项涉及密码的使用期限，已在“密码时效”一节讲过。

在 Linux 主机上，有关组的信息被存储在/etc/groups 文件里。列表 4-13 显示了来自这个文件的一个例子。

列表 4-13 /etc/groups 文件的例子

```
root:x:0:root
ataylor:x:501:finance,printing
```

/etc/group 文件的构造很像/etc/passwd 文件，数据项都是用冒号隔开。这个文件可以分解为一个组名、一个密码、它的 GID 和一个用逗号分隔的组员列表。

```
groupname:password:GID:member,member
```

group 文件里的密码使用户可以使用 newgrp 命令登入那个组。如果启用了隐蔽密码，那么就与 passwd 文件一样，group 文件里的密码用一个 x 代替，而真实的密码被存储在/etc/gshadow 文件里。

补充知识：登录消息

登录屏是用户看到的第一个画面。在这个画面上放一些重要的警告和信息是一个好主意。为此，你需要编辑/etc/issue 和/etc/issue.net 文件的内容。issue 文件在通过主机控制台上的命令行登入时显示，而 issue.net 文件在通过 SSH 会话登入到命令行时显示。大部分发行版使用这些文件都是这个目的，包括 Red Hat 和 Ubuntu。这些文件可以包含普通文字和特殊扩展字符（这些字符允许输出颜色、换行和回车等）的组合。

你还应该包含一个警告信息，声明禁止对主机未经授权的访问并且会起诉。你可以使用这些文件里的扩展字符和主机上的数据构建登录界面。推荐使用如下的登录消息。

```
^[c
\d at \t
Access to this host is for authorized persons only.
Unauthorized use or access is regarded as a criminal act
```


and is subject to civil and criminal prosecution. User activities on this host may be monitored without prior notice.

^[\c 扩展字符清空屏幕，而\ld 和\lt 扩展字符分别显示主机当前日期和时间。如果去了解 issue、issue.net 和 getty 的 man 页，还可以利用其他扩展字符。

除了/etc/issue 和/etc/issue.net 文件之外，/etc/motd 文件的内容紧跟在命令行登录之后显示，你可能希望调整它们以便把一个可接受使用策略（Acceptable Use Policy）或类似的信息包含进来。

4.2.10 配置 shell 和环境

用户通过验证和授权之后，他的 shell 就开启了。大多数 shell 都可定制，而且许多用户最终都调整了 shell 环境的各个方面以帮助他们更快速、更高效地工作。

Bash shell 从/etc/profile 文件读取它的初始配置。该文件通常包含对其他全局配置文件的引用，被用来为主机上 root 用户之外的所有用户配置 Bash。最后处理用户主目录里的配置文件。 .bash_profile 和 .profile 文件最常用，每个用户的主目录里都有这些文件。

■注：查看 bash man 页的 INVOCATION 部分可以找到其他配置文件的完整列表。

1. 环境变量

之所以定制 shell，其中一个主要原因是为了设置环境变量。这些变量作为默认选项被许多应用程序使用。它们可以定义一些特征，如偏好的文本编辑器、偏好的语言以及用 ls 列出文件和目录时所使用的颜色。你还可以用自己的脚本定义自己要用的变量。

要得到所有环境变量的完整列表，请使用 env 命令。表 4-6 列出了最常定制的变量。

表 4-6 环境变量

名字	用途
HOME	用户的主目录
LANG	定义应用程序要使用的语言
LS_COLORS	定义 ls 命令要使用的颜色
MAIL	用户邮箱的位置
PATH	一个用冒号分隔的目录列表，shell 在这些目录里查找可执行文件
PS1	定义标准的提示符
SHELL	当前的 shell
-	把本次会话中执行的上一个命令包含进来

通过 echo 命令可以显示一个环境变量的内容。在所希望显示的变量名前加\$。

```
$ echo $PS1
\u@\h:\w$
```

上述输出是一个特殊扩展码字符串，它显示了用户名\u、主机名\h、当前工作目录\w 和最后的字符\$。如果提示符以 root 用户身份显示，\ \$就显示为一个英镑（#）符号，否则显示

为美元符号 (\$)。至于可用扩展码的完整列表，请看 `bash man` 页的 `PROMPTING` 部分。

环境变量可以通过在 `Bash` 配置文件里定义或者从命令行设置来更改。如果希望改变提示符以使之包括一个时间戳并提供更多键入命令的空间，可以添加 `\T` 和 `\n` 代码。

```
$ PS1="[\T] \u@\h:\w\n\$ "
[12:50:59] jsmith@au-mel-ubuntu-1:~
$
```

■提示：你可能已经注意到\$有时使用，有时不使用。这里的规则很简单，如果涉及到变量并给它加了前缀\$，那么所关注的就是变量的值（即变量的内容）。如果没有\$，就是在讨论变量本身。

另外一个有用的例子是往路径中添加目录。可以很快地把目录添加到路径前面或后面。如下所示就可以把单个目录添加到路径的开头。

```
$ PATH=/home/ataylor/scripts:$PATH
```

这里把目录 `/home/ataylor/scripts` 添加到了路径的前面，然后通过用冒号分隔并指定 `$PATH` 值把已有的路径包括进来。这样可以把二进制文件、脚本或其他应用程序放到路径里，从而每次运行一个命令或应用程序时都会搜索它。在本例中，当执行程序时，Linux 在搜索任何别的地方之前首先会在 `/home/ataylor/scripts` 目录中寻找这个命令。

你可以使用同样的基本构造法在路径末尾添加一个目录。

```
$ PATH=$PATH:/home/ataylor/scripts
```

那么在运行一个命令时，Linux 会搜索路径里的所有目录，如果没有找到匹配的命令或应用程序，它会搜索附加在后面的目录。

`Bash` 会把任何 `KEY=value` 类型的字符串认作是对环境变量的赋值。变量大写是约定的写法。

当然，在命令行上设置环境变量只在会话期间改变它们。如果退出，它们会还原到以前的配置。要让这样的改动持久，请把它们放在位于主目录的 `.bash_profile` 文件里，如下所示。

```
PATH=$PATH:/home/ataylor/scripts
export PATH
```

这里指定了新路径，然后使用了一个特殊命令 `export` 传送这个改动。通常对环境变量的改动只在当前所在的会话或脚本起作用。为了在别的会话或脚本里使用它们，需要导出它们。要使一个路径或其他环境变量的改动为所有用户使用，请把改动添加到 `/etc/profile` 文件。该文件被所有用户（除了 `root` 用户；要修改 `root` 用户的变量，请使用 `/root` 目录里的 `.bash_profile` 文件）用来设值。

■提示：在 <http://tldp.org/HOWTO/Bash-Prompt-HOWTO/> 上可以找到更多有关配置 `Bash` 提示符的信息。

2. 命令别名

配置 `shell` 的第二个原因是为了创建目录别名。使用别名可以为常用的命令创建快捷方式或设置默认选项。最好的例子就是 `rm` 命令在许多发行版中都默认开启的一个别名，第 3 章提到过它。

用 `rm` 删除一个文件时，如果不传递 `-i` 选项，它就不会要求确认。通过使用别名，你可以让 `shell` 在每次键入 `rm` 时执行 `rm -i`，这样在删除文件时就总会有确认提示。通过 `alias` 命令创建一个别名，然后创建并删除一个文件。

```
$ alias rm='rm -i'
$ touch test
$ rm test
rm: remove regular file `test'? y
```

通过把它添加到主目录的 `.bash_profile` 配置文件可以使别名持久。

为了得到 `shell` 里定义的所有别名列表，要不带任何参数地运行 `alias` 命令。

```
$ alias
alias rm='rm -i'
alias v='ls -lt'
alias r='ls -lrt'
```

这里定义了交互式删除命令的别名和另外两个别名，在按照修改日期的顺序列出拥有大量文件的目录时，这两个别名会节省键入次数。

如果不是在设置默认选项，那么定义的别名不要和已有命令同名。通过指定可执行文件的完整的路径，你仍然可以运行原始的命令。但是它可能会在你没有注意到的地方产生意外的效果（例如在自动运行的脚本里）。

要删除一个别名，请使用 `unalias` 命令。要移除交互式删除命令，使用下面这个命令。

```
$ unalias rm
```

要阅读有关别名的更多内容，请看 `bash man` 页的 `ALIASES` 部分。

`Bash shell` 非常强大和灵活，它可以使日常的管理工作非常容易。如果想了解有关 `Bash` 的更多内容以及它的更多功能，请参阅 <http://www.tldp.org/LDP/Bash-Beginners-Guide/html/> 和 <http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html>。

4.3 控制对主机的访问

有相当多的用户特性都可以控制，包括用户什么时间或如何登入，他们的密码看起来怎么样，以及他们多久更改和重置一次密码。这些管理任务都会用户在登入主机时进行，一般是由一系列模块来做。这些模块以可插入鉴别模块（`Pluggable Authentication Modules, PAM`）为大家所知。几乎所有的 `Linux` 发行版，包括 `Red Hat` 和 `Ubuntu`，都靠 `PAM` 控制用户如何以及何时与主机交互。

本节介绍 `PAM` 及其工作方式。一般不会对 `PAM` 配置更改很多，但重要的是理解它是如何工作的。

■注：第 16 章将对 `PAM` 如何与别的身份验证机制一起使用（例如与 `Active Directory` 和 `LDAP` 的集成）的问题进行详细讲解。

PAM 最初是 Sun Microsystem 公司为了提供一个插入式身份验证框架而设计的。它在 Linux 领域被大量使用和开发，并且存在大量执行多种功能的 PAM 模块，范围从密码检查到主目录的创建。PAM 模块最初被用来为缺乏身份验证或某种身份验证功能的应用程序提供身份验证和其他服务。之后，随着更完善的身份验证类型的出现，PAM 就变成一种集成和扩展身份验证机制的方式。不必为新的身份验证方法重写每个程序，所需的一切就是添加 PAM 支持程序。然后 PAM 通过一个标准的 API 承担身份验证这个重活。

配置 PAM

本质上 PAM 是一个层次化的结构，涉及身份验证和应用程序要执行某个动作时所执行的授权检查。这些检查任务被集中放在栈里。例如，在登入时检查用户是否存在，接着检查用户的密码是否有效以及密码是否到期。这个栈通常由多个 PAM 模块组成，每个模块执行某项验证功能。此外，有些检查必须要通过（例如，用户必须存在），而其他的检查可能是可选的。理解 PAM 的最好方式是分析一些 PAM 配置文件。

在大部分 Linux 发行版上，在两个可能的位置可以找到 PAM 配置信息。历史遗留的文件 `/etc/pam.conf` 在过去用来保存 Linux 发行版上的 PAM 配置信息，但是现在一般反对使用它，取而代之的是 `/etc/pam.d` 目录。大部分现代版本的 Red Hat 和 Ubuntu 用这个目录为许多具有针对 PAM 设置的服务保存配置文件。服务和为验证它而设计的程序使用同一个名字；例如 `passwd` 命令的 PAM 配置包含在一个 `/etc/pam.d/passwd` 文件里。这些文件被称为服务配置文件。

有许多种服务配置文件——如用户登入主机时所使用的一个叫作 `login` 的程序。这个登录程序在用户登录时被触发，在 `pam.d` 目录里面可以找到一个包含此程序身份验证配置且名为 `login` 的文件。还可以找到一个叫作 `sshd` 的文件，它对通过 SSH 连接登入的用户执行类似的任务。

在 `/etc/pam.d` 目录里可以找得到 PAM 默认配置自带的常用服务、`passwd` 命令和定时任务（`cron`）调度的后台程序。在这些文件里可以找到这些程序所使用的身份验证配置。

■注：第 5 章将讲述定时任务的设置（`crontab`）以及如何调度任务和动作。

不过，我们不打算研究每个具体的文件，因为这些服务大部分都依赖某个共同的身份验证配置。Red Hat 和 Ubuntu 两者都有独立的文件定义这个共同的身份验证配置。许多服务文件引用并包含这个共同的配置。在 Red Hat 上，这个文件是 `/etc/pam.d/system-auth`，它在主机安装时自动生成并通过一个叫作 `authconfig` 的特别命令更新。在 Ubuntu 上，同样的任务由 4 个独立文件执行：`common-auth`、`common-password`、`common-session` 和 `common-account`。让我们从列表 4-14 看看 Red Hat 上 `system-auth` 文件的内容。

列表 4-14 login 的 PAM 文件

```
%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.
auth required pam_env.so
auth sufficient pam_unix.so try_first_pass nullok
```



```
auth    required    pam_deny.so

account required    pam_unix.so

password required    pam_cracklib.so try_first_pass retry=3
password sufficient    pam_unix.so try_first_pass use_authtok nullok md5 shadow
password required    pam_deny.so

session optional    pam_keyinit.so revoke
session required    pam_limits.so
session [success=1 default=ignore] pam_succeed_if.so service in crond quiet use_uid
session required    pam_unix.so
```

`system-auth` 和其他服务配置文件有 4 个可能的指令。让我们仔细分析列表 4-14 里的一行。

```
auth    sufficient    pam_unix.so try_first_pass nullok
```

这行的第一个指令是 `auth`，它是正要配置的管理组。在 PAM 里有 4 个管理组，它们代表可配置的身份验证和授权进程的不同部分。

- `auth`: 这些模块执行用户的身份验证，例如，检查密码。
- `account`: 这个管理组处理账号验证任务，例如，确认用户账号已开启或者是否只有 `root` 用户可以执行某个动作。
- `password`: 这些模块设置密码，例如，检查密码以确保它足够强壮。
- `session`: 这些模块检查、管理和配置用户会话。

通常每个管理组都分配了一个以上的模块，这些模块一般按照指定的顺序检查，每个模块会返回一个成功或失败的结果。一个特定的模块可能在一个 PAM 配置里被不止一次地指定。例如，从列表 4-14 可以看到所有 4 个管理组里都指定了 `pam_unix.so` 模块。

```
auth        sufficient    pam_unix.so try_first_pass nullok
account     required      pam_unix.so
password    sufficient    pam_unix.so try_first_pass use_authtok nullok md5 shadow
session     required      pam_unix.so
```

这表示承担大部分标准 Unix 身份验证功能（如输入一个传统的密码）的 `pam_unix.so` 模块可以执行每个管理组的检查任务和函数。例如，它可以确认用户的密码在 `auth` 组里是正确的，还可以为 `account` 组确认用户的存在性。

接下来的指令 `sufficient` 被称为控制标志，它告诉 PAM 如何处理模块的结果。如前所述，有些检查比其他的更重要。控制标志告诉 PAM 如何处理成功或失败的结果以及那个结果如何影响整个验证过程。表 4-7 列出 4 个 PAM 控制标志。

表 4-7 PAM 的控制标志

标志	说明
required	为了验证成功，一个必需的（required）模块必须成功
requisite	如果一个必需的（requisite）模块失败了，那么验证直接失败
sufficient	如果此模块成功，那么验证直接成功
optional	此模块的成功或失败不影响验证结果

required 标志意思是为了验证过程成功，这个模块结果必须是成功。如果此模块结果是失败，那么整个验证过程也是失败。如果栈里超过一个模块，那么该栈中的其他模块也要被处理，但整个验证还是失败。

requisite 标志也表示为了验证成功，这个模块结果必须是成功。不过与 **required** 标志不一样的是，这个模块的成功或失败会立刻通知给请求身份验证的服务，并且验证过程会结束。这意味着如果所有模块集中放在栈里并且一个带 **requisite** 控制标志的模块失败了，那么剩下待处理的这些模块将不会被执行。相比之下，如果这个模块带的是 **required** 控制标志，那么剩下的模块会继续被处理。

下一个控制标志是 **sufficient**。**sufficient** 标志的意思是这个模块的成功是验证过程成功的充分条件，或者说，如果模块都放在栈里，那栈就是成功的。这取决于在这个模块之前没有别的 **required** 模块失败。但是，如果一个 **sufficient** 模块失败了，整个栈并不失败。

最后一个控制标志是 **optional**。一个 **optional** 模块对整个验证过程或模块栈的成功或失败不重要。它的成功或失败不会决定整个验证过程的成功或失败。

接下来的指令 **pam_unix.so** 表示要用的 PAM 模块和它的位置。如果你指定了一个 PAM 模块而没有路径，那么这个模块被假定位于 **/lib/security** 目录。你还可以通过在这里给它提供路径来指定一个其他位置的模块，正如下面一行所示。

```
auth required /usr/local/pamlib/pam_local.so id=-1 root=1
```

最后的指令是要传递给 PAM 模块的参数——本例给 **pam_unix.so** 模块传递了参数 **try_first_pass** 和 **nullok**。**try_first_pass** 参数告诉该模块看看它是否已经接收一个密码，如果是，就使用那个密码做身份验证。**nullok** 参数告诉该模块可以是空密码。大多数模块会忽略传递给它们的无效或不正确参数，虽然有些模块会产生一个错误消息或失败，但是该模块会继续被处理。

■ **提示：**在 Ubuntu 上可以找到大多数 PAM 模块的 man 页（例如，**man pam_unix** 会返回 **pam_unix** 的 man 页）。在 Red Hat 上可以在 **/usr/share/doc/pam-version** 目录里查找文档。还可以在 <http://www.kernel.org/pub/linux/libs/pam/> 上查找文档。

需要提及的最后一个 PAM 函数是 **include**。使用 **include** 函数可以把一个文件包含于另一个文件。这是把共同的配置包含于具体服务配置文件的方式。为了看到这个函数，让我们看看 Ubuntu 上 login PAM 服务配置文件的一个片段，如列表 4-15 所示。

列表 4-15 Ubuntu 上 login PAM 服务配置文件

```
# Standard Unix account and session
@include common-account
@include common-session
@include common-auth
@include common-password
```

通过使用 **@include** 格式，你可以在一个 PAM 服务配置文件里包含其他文件。因此，**@include common-account** 会把文件 **common-account** 的内容包含于 **login** 文件。在那个文件里指定的每个模块现在都会在使用 **login** 文件时得到处理。该文件在包含它的地方得以展开和分析，并且按顺序执行那个被包含文件中的所有模块。

你还可以把 `include` 选项用作一个控制标志，如下所示。

```
auth include system-auth
```

它会把所有来自文件 `system-auth` 的 `auth` 类型的行包括进来。

4.4 sudo 命令详解

正如本章较早所讲的那样，`sudo` 命令允许利用另一个用户（大多数情况下是 `root` 用户）的特权运行一些命令。该命令的工作方式很像 Microsoft Windows 里的 `RunAs` 命令，`RunAs` 命令允许一个用户以另一用户的身份运行一个命令。

■注：另一个叫作 `su` 的命令，也就是大家所知的替代用户或切换用户的命令，允许从一个用户改到另一个用户而无需登入、退出。它常用于切换到 `root` 用户执行某个动作。通过它的 `man` 页可以阅读相关信息。注意 `su` 在 `root` 用户锁定时不能用，在 Ubuntu 上就是这样。可以通过为 `root` 用户设置密码为账号解锁。

为了使用该命令，你应该键入 `sudo` 和要执行的那个命令。如果可以使用 `sudo`，它会提示输入一个密码，通常就是你自己的用户密码，然后指定的命令会以 `root` 用户身份执行。这样，不必真正以 `root` 用户身份登入就可以执行 `root` 用户才能做的动作，比如创建用户。从列表 4-16 可以看到 `sudo` 命令的运行情况。

列表 4-16 使用 `sudo`

```
$ sudo userdel ataylor
We trust you have received the usual lecture from the local System Administrator.
It usually boils down to these three things:
    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.
Password:
```

这个相当吓人的消息一般出现在第一次使用 `sudo` 命令时，以后就只得到输入密码的提示。

`sudo` 命令不是在每次使用时都提示输入密码。输入密码之后，`sudo` 命令会给一个宽限期，在宽限期内它不会提示输入密码。这个期限在 Red Hat 上是 5min，在 Ubuntu 上是 15 min。过了这个期限再运行 `sudo` 命令时，它会再次提示输入密码。

Ubuntu 和 Red Hat 都默认安装 `sudo` 命令。事实上，Ubuntu 甚至不为 `root` 用户设置密码，它宁愿鼓励一直使用 `sudo` 来运行特许命令。在 Ubuntu 上，`admin` 组的任何成员都可以运行 `sudo` 命令。在 Red Hat 上，我们较早前配置过它，因此只要是 `wheel` 组的成员，就可以使用 `sudo` 来运行命令。

`sudo` 命令还是可配置的。你可以精确指定可使用 `sudo` 来执行的命令，包括合起来作为命令类别的命令组。你可以配置 `sudo` 命令使它允许用户执行所有命令、部分命令或者甚至不要求输入密码就允许执行命令（但是不推荐）。

■提示：正如本章开头所讲的那样，通过使用一个叫做 `visudo` 的特别命令编辑一个叫 `/etc/sudoers` 的文件，你可以精确配置 `sudo` 可做的事。“配置 `sudo`”一节将对此做更多的解释。

那么如果未得到执行 `sudo` 命令的许可，会发生什么事？在列表 4-17 中尝试以用户 `ataylor` 的身份使用 `sudo` 命令，而该用户没有使用 `sudo` 命令的授权。

列表 4-17 未授权的 `sudo` 命令

```
$ sudo useradd -m -c 'Illegal User' iuser
ataylor is not in the sudoers file. This incident will be reported.
```

使用 `sudo` 命令的失败尝试会被主机的 `syslog`（或系统日志）服务记录，然后这个消息被发送到 `/var/log` 目录下的一个文件里。在基于 Red Hat 的发行版上，可在 `/var/log/secure` 文件里看到 `sudo` 命令失败记录。在 Ubuntu 上。这些记录出现在 `/var/log/auth.log` 文件里。它会产生如下这样一条记录消息，显示日期、时间、试图执行 `sudo` 命令的用户和该用户试图执行的那个未授权命令。

```
Sep 1 20:27:43 au-mel-rhel-1 sudo:    ataylor : user NOT in sudoers ; TTY=pts/1 ;
PWD=/home ; USER=root ; COMMAND=/usr/sbin/useradd -m -c 'Illegal User' iuser
```

利用这些信息可以监视那些在主机上试图执行不合适动作的人，而且可以用来检测企图破坏安全的行为。

■注：第 18 章将详细讲述记录 and 如何监视像本节所述的这种消息，以及如何发送警告或采取某种行动。

配置 `sudo`

`sudo` 命令查看 `/etc/sudoers` 文件以验证命令的运行授权。配置 `sudoers` 文件可以限制对特定用户、特定命令以及在特定主机上的访问。

让我们看看 `/etc/sudoers` 文件。首先，你需要使用 `visudo` 命令编辑 `/etc/sudoers` 文件。`visudo` 命令是一个为了与 `sudo` 命令一起使用而设计的特殊编辑器，它是编辑 `sudoers` 文件最安全的方式。此命令把该文件锁定以防止多方同时编辑，提供基本的完好性检查，并检查语法错误。如果 `/etc/sudoers` 文件目前正在被编辑，你就会接收到稍候再试的消息。

我们从研究如何让用户 `ataylor` 运行 `userdel` 命令开始。把列表 4-18 中的内容添加到了 `sudoers` 文件。

列表 4-18 `sudoers` 样例

```
ataylor ALL=/bin/userdel
```

这一行可以分解成几部分。

```
username host = command
```

列表 4-18 显示用户 `ataylor` 被允许在所有主机上（使用了 `ALL`）像 `root` 用户一样使用命令 `/bin/userdel`。在命令选项里指定的任何命令都必须用它的完整路径定义。你还可以指定一个以上被授权使用的命令，每个命令用逗号隔开，如下一行所示。

```
ataylor ALL=/bin/userdel,/bin/useradd
```

在上一行中，`ataylor` 马上被授权可像 `root` 用户一样使用 `userdel` 和 `useradd` 命令。`sudoers`

文件里的所有配置行都只能在一行里，可以使用\表示配置在下一行继续进行。

单个 `sudoers` 文件为配置多个主机而设计。因此，它允许针对主机的访问控制。你可以在一个中心主机上维护 `sudoers` 文件并向所有的主机分发更新文件。

■注：第 19 章将讲述配置的管理以及如何把这个文件分发给多个主机。

有了主机访问控制，你就可以为不同的主机定义不同的授权，如列表 4-19 所示。

列表 4-19 在多个主机上使用 sudo 授权

```
ataylor au-mel-rhel-1=/bin/userdel,/bin/useradd
ataylor au-syd-ubuntu-1=ALL
```

在列表 4-19 中，用户 `ataylor` 在主机 `au-mel-rhel-1` 上只允许使用 `userdel` 和 `useradd` 命令，而在主机 `au-syd-ubuntu-1` 上，如 `ALL` 选项表示的那样，她被允许使用所有的命令。

■注：要谨慎使用 `ALL` 变量定义对主机上所有命令的访问权限。`ALL` 变量没有授权配置的粒度区分。

通过对一个特定目录里的命令给予访问授权，你可以稍加选择性地授权。

```
ataylor au-mel-rhel-1=/bin/*
```

这只适用于指定的目录，不适用它的任何子目录。例如，如果授权对 `/bin/*` 目录的访问，那么就不能运行 `/bin/extra/` 目录里的任何命令，除非明确定义了对那个目录的访问，如下一行的配置。

```
ataylor au-mel-rhel-1=/bin/*,/bin/extra/*
```

有时希望给一个用户访问某个特定命令的授权，但希望那个命令以另一个用户的身份运行。比如说，需要以特定用户身份开启和停止一些后台程序，如 `MySQL` 或 `named`。你可以把用户名放在该命令前面的括号里，以此来指定希望该命令运行时的用户身份，如下所示。

```
ataylor au-mel-rhel-1=(mysql) /usr/bin/mysqld,(named) /usr/sbin/named
```

可以想象，被授权的命令、用户和主机列表可以变得非常长。`sudo` 命令还带定义别名的选项。别名是许多类似的用户、命令和主机的集合。一般别名的定义在 `sudoer` 文件的开头进行。

让我们看看一些别名。第一种别名是 `User_Alias`，它聚合类似的用户。

```
User_Alias ADMIN = ataylor,jsmith
```

用正在使用的别名类型（在本例中就是 `User_Alias`）定义一个别名，跟着是定义的别名名字，在这里是 `ADMIN`。接下来指定属于这个别名的用户列表。然后就可以在配置文件里引用这个别名了。

```
ADMIN=/bin/userdel,/bin/useradd, \
(named) /usr/sbin/named
```

上一行规定别名 `ADMIN` 里的用户可以使用命令 `userdel`、`useradd` 和 `named`。

可定义的下一类型的别名是命令别名 `Cmnd_Alias`，它聚合命令。

```
Cmnd_Alias USER_COMMANDS = /bin/userdel,/bin/useradd
```


这个别名可以和刚创建的用户别名一起使用。

```
ADMIN ALL=/bin/groupadd,USER_COMMANDS
```

现在定义在别名 `ADMIN` 里的所有用户都可以在所有主机上使用命令 `/bin/groupadd` 以及在命令别名 `USER_COMMANDS` 里定义的所有命令。

你还可以指定一个聚合主机的别名。`Host_Alias` 别名可以指定主机名、IP 地址和网络的列表。

```
Host_Alias SERVERS = au-mel-rhel-1, au-mel-rhel-2, au-syd-rhel-1
```

这个别名可以与前面定义的别名一起使用。

```
ADMIN SERVERS=USER_COMMANDS
```

现在 `ADMIN` 别名里指定的所有用户都可以在 `SERVERS` 别名组里规定的主机上运行 `USER_COMMANDS` 里指定的命令。

你还可以通过在别名前放一个感叹号 (!) 来排除它们。让我们看一个这样的例子。首先，用希望用户使用的一些命令定义一个命令别名，然后可以结合一个 `sudo` 配置行使用那个别名。

```
Cmnd_Alias DENIED_COMMANDS = /bin/su,/bin/mount,/bin/umount
ataylor au-mel-rhel-1=/bin/*,!DENIED_COMMANDS
```

在这里，用户 `ataylor` 可以使用除了在 `DENIED_COMMANDS` 别名里定义的命令之外的 `au-mel-rhel-1` 主机上 `/bin` 目录下的所有命令。

你还可以使用其他方式授权用户使用 `sudo` 命令，让我们看看其中一种方式。在 `sudoers` 文件里面，你可以基于主机上的组信息定义另一类型的别名，方式就是在组名前加前缀 %。

```
%groupname ALL=(ALL) ALL
```

然后把 `groupname` 替换为你主机上定义的某个组名。这意味着所定义的组里的所有成员都可以执行授权给他们的任何命令，在本例中就是所有主机上的所有命令。

在 Red Hat 主机上，一个叫作 `wheel` 的组就是这种用途。如果把 Red Hat 主机上 `/etc/sudoers` 文件中如下一行的注释符号去掉，那么被添加到 `wheel` 组的所有用户就都可以使用 `sudo` 命令，从而获得主机上 `root` 用户的特权。在 Ubuntu 上这个组被称为 `admin` 而不是 `wheel`。

```
%wheel ALL=(ALL) ALL
```

另外，`sudoers` 文件本身有许多选项和默认设置可以定义，以改变 `sudo` 命令的行为。例如，你可以配置 `sudo` 让它在有人使用 `sudo` 命令时发送电子邮件。要规定向谁发送那封邮件，你可以使用下面一行里的选项。

```
mailto "admin@au-mel-rhel-1.yourdomain.com"
```

然后你可以使用别的选项进一步修改 `sudo` 发送那封邮件的时间。

```
mail_always on
```

为了解可用的默认设置和选项的种类，表 4-8 详细说明了与该邮件相关的选项列表。

表 4-8 sudo 运行时发送电子邮件的选项	
选项	说明
mail_always	每次用户运行 sudo 时发送电子邮件。这个标志默认设置为 off
mail_badpass	如果运行 sudo 的用户没有输入正确的密码就发送电子邮件。这个标志默认设置为 off
mail_no_user	如果运行 sudo 的用户在 sudoers 文件里不存在，就发送电子邮件。这个标志默认设置为 on
mail_no_host	如果运行 sudo 的用户存在于 sudoers 文件中但没有被授权在本机运行命令，就发送电子邮件。这个标志被默认设置为 off
mail_no_perms	如果运行 sudo 的用户存在于 sudoers 文件中但没有他试图运行的这个命令的授权，就发送电子邮件。这个标志被默认设置为 off

sudoers 的 man 页详细说明了許多其他选项和默认设置。
sudo 命令本身还有一些命令行选项可一起使用。表 4-9 给出了一些最有用的选项。

表 4-9 sudo 的命令行选项	
选项	说明
-l	在当前主机上打印当前用户被允许（和被禁止）运行的命令列表
-L	列出在 sudoers 文件里所设置的全部默认选项
-b	在后台运行给定的命令
-u user	以一个非 root 用户的身份运行指定的命令

-l 选项特别有用，它可以确定当前主机上当前用户被授权和被禁止运行的命令。

```
$ sudo -l
Password:
User ataylor may run the following commands on this host:
    (root) ALL
```

sudo 命令很复杂，如果实现得不合适，就可能给主机打开一个安全缺口。推荐在实现它之前谨慎地测试所有 sudo 配置，并且详尽地探究 sudo 和 sudoers 的 man 页。

补充知识：谁正在干什么？

清楚你的用户正在干什么 是用户管理的一个重要部分。第 18 章将介绍日志。的确，为了清楚用户的动作，首先要利用的资源之一就是日志文件的内容。当然别的命令和资源对此也是有用的。

who 命令显示当前登录主机的所有用户以及他们登录的终端。如果用户是远程连接，该命令会显示他们连接所用的 IP 地址或主机名。

```
$ sudo who
root      tty1      Jul  3 12:32
ataylor   pts/0     Jul  8 11:39 (host002.yourdomain.com)
```

你可以修改 who 命令的输出。在 who 的 man 页可以看到这些选项的完整列表。可能最有用的命令行选项是-a，它组合了各种命令行选项，提供了一个对登录用户、登录进程以及主机重启与运行级别资料的详细概述。

有用的命令还有 last 和 lastb，它们分别显示一个用户上次登录主机的时间记录和一个错

误用户登录的记录。如果不带任何选项地执行 last 命令，它会打印一个最近的登录报告。

```
$ sudo last
root      tty1                Sat Jul 3 12:32  still logged in
ataylor   pts/0          192.168.0.23    . Sat Jul 3 14:25 - 14:26 (00:01)
reboot    system boot  2.4.20-28.8     Sat Jul 3 12:31    (4+05:40)
```

可以看到，last 命令告诉你 root 用户登录过并且还在主机上。这个列表还显示了用户 ataylor，她从 IP 地址 192.168.0.23 登入并停留了 1 秒。最后一项显示了一个 reboot 项。主机每次重启都会记录一个输入项，以便提供重启的时间和主机启动进入的内核版本。

lastb 命令产生同样类型的报告，但只列出那些失败的登录记录。换句话说，它列出那些因输入不正确的密码或者一些别的错误而导致登录失败的记录。

与 last 和 lastb 相关的命令是 lastlog。lastlog 命令显示一个展示主机上所有用户（包括那些从未登入的用户）登录状态的报告。它显示一个所有用户和他们上次登录的日期和时间的列表，或者显示一个 “**Never Logged In**” 的消息，如果那个用户从未登入的话。使用命令行选项可以在记录中搜索特定用户。请阅读 lastlog 命令的 man 页以了解进一步的信息。

4.5 小 结

在本章中，你学习了如何从命令行或者通过 GUI 界面创建用户和组，还了解了登入主机时发生的事情，以及有关 PAM 的知识和如何控制对主机的访问。

本章详细介绍了 sudo 命令以及如何利用它避免使用 root 用户管理主机。另外，分析了如何配置 sudo 来控制谁可以访问特定的命令以及如何报告 sudo 的使用情况。最后介绍了一些监视用户登录的方式。

下一章将介绍主机启动时发生的事情以及如何开启、停止和管理服务。

第 5 章 启动与服务

在前几章，你学习了如何安装 Linux 主机，了解了一些基本的 Linux 概念以及用户和组的概念。本章将更深入地研究 Linux 主机的运转方式并分析它在“后台”的工作方式。

本章将介绍主机开始运转或叫启动（boot）时发生的事情。我们将一步进入这个过程，讲解如何在各种模式下启动主机以及如何配置和修改启动进程。为了说明所有这些，本章将阐述主机从基本的输入输出系统（BIOS）启动到登入这整个过程。

除了启动过程之外，本章还会介绍主机如何开启和停止应用程序、系统服务和其他进程。Red Hat 和 Ubuntu 每个发行版对这些服务的添加、移除、开始和停止等管理都稍微有点不同。我们会说明每个发行版对这些服务的管理方式，包括它们的细微差别。你将学习什么是服务，如何开启和停止服务，以及如何查看它们的状态。最后会讲解如何在主机启动或关闭时让服务自动开启和停止。

5.1 当主机启动时发生了什么事？

启动过程（boot 是 bootstrap 的简写）通常包括 3 个独立而又互相关联的过程：BIOS、引导程序和操作系统的载入。

1. BIOS 初始化并检查硬件。
2. 引导程序要求选择一个操作系统来加载。
3. 最后，加载并初始化操作系统。

这些步骤不只针对 Linux；你可以发现大部分操作系统执行类似的一套功能和步骤。

5.1.1 BIOS

让我们更深入地看看主机启动时发生的事情。你可能已经注意到，当打开主机时，通常会听到几次嘟嘟声和呼呼声，并在前端面板和键盘上看到闪烁的光。这是主机启动的第一步，这个过程由主板上一个叫作 BIOS 的小芯片控制。BIOS 执行初步的系统检查或上电自检（power-on-self-test, POST）操作，看看附带在系统上的硬件（例如内存、硬盘驱动器、键盘

和显卡) 的不同比特位是否可用。可以根据 BIOS 改变不同的设置，不过我们把它留给读者有空时去研究。

BIOS 还会查询其他硬件，如硬盘控制器，并初始化它们的板上芯片。接下来根据硬件的不同，BIOS 屏幕会显示已找到控制器的设备信息。通过按一组键（通常在硬件初始化时显示在屏幕上），你可以得到一个配置控制器或其他硬件的选项。通过这个菜单你可以操作主机的配置；例如，使用它可以在硬盘控制器上设置 RAID 或者检修现有硬件配置的问题。

■注：改变某些配置可能有危险，例如不正确地更改一个 RAID 配置可能会损坏数据，因此要小心使用这些菜单。

BIOS 还允许修改主机的启动源。启动源就是主机从中寻找操作系统的媒介，如硬盘驱动器。启动次序设置允许从几个源之一启动：硬盘驱动器、CD/DVD 或者 USB 便携存储设备。默认情况下，主机通常会尝试从附带的硬盘驱动器启动。如果配置过的话，它会寻找备选源，如 CD/DVD 驱动器或者 USB 便携存储设备。

不同的主板制造商打开启动源菜单的方式不同，例如，可能是按 Esc、Del 或者一个像 F1、F2 或 F10 这样的功能键。通常在屏幕上会出现按键信息。通过按适当的键通常会显示一个菜单，可以选择想要的启动次序。

5.1.2 引导程序

BIOS 利用启动源设置指定寻找启动程序下一阶段——引导程序的位置。BIOS 用硬盘驱动器上的一个叫作主引导记录 (master boot record, MBR) 的特别扇区告诉它要启动什么引导程序。MBR 有 512 字节长，包括几项数据。还记得第 2 章所讲的如何为磁盘分区以及安装 GRUB 引导程序吗？有关如何对磁盘分区（即分区份数、每份的大小和它们的名字）的信息被写入 MBR。这占用了记录的前 64 字节。接下来，安装进程往 MBR 里添加 GRUB 引导程序，此程序占用了剩下的大部分字节。

图 5-1 显示了一个典型的硬盘配置，可以看到它被划分为 MBR 和数据分区。

■注：关于对硬盘结构的讨论，请参阅 <http://www.ranish.com/part/primer.htm>。

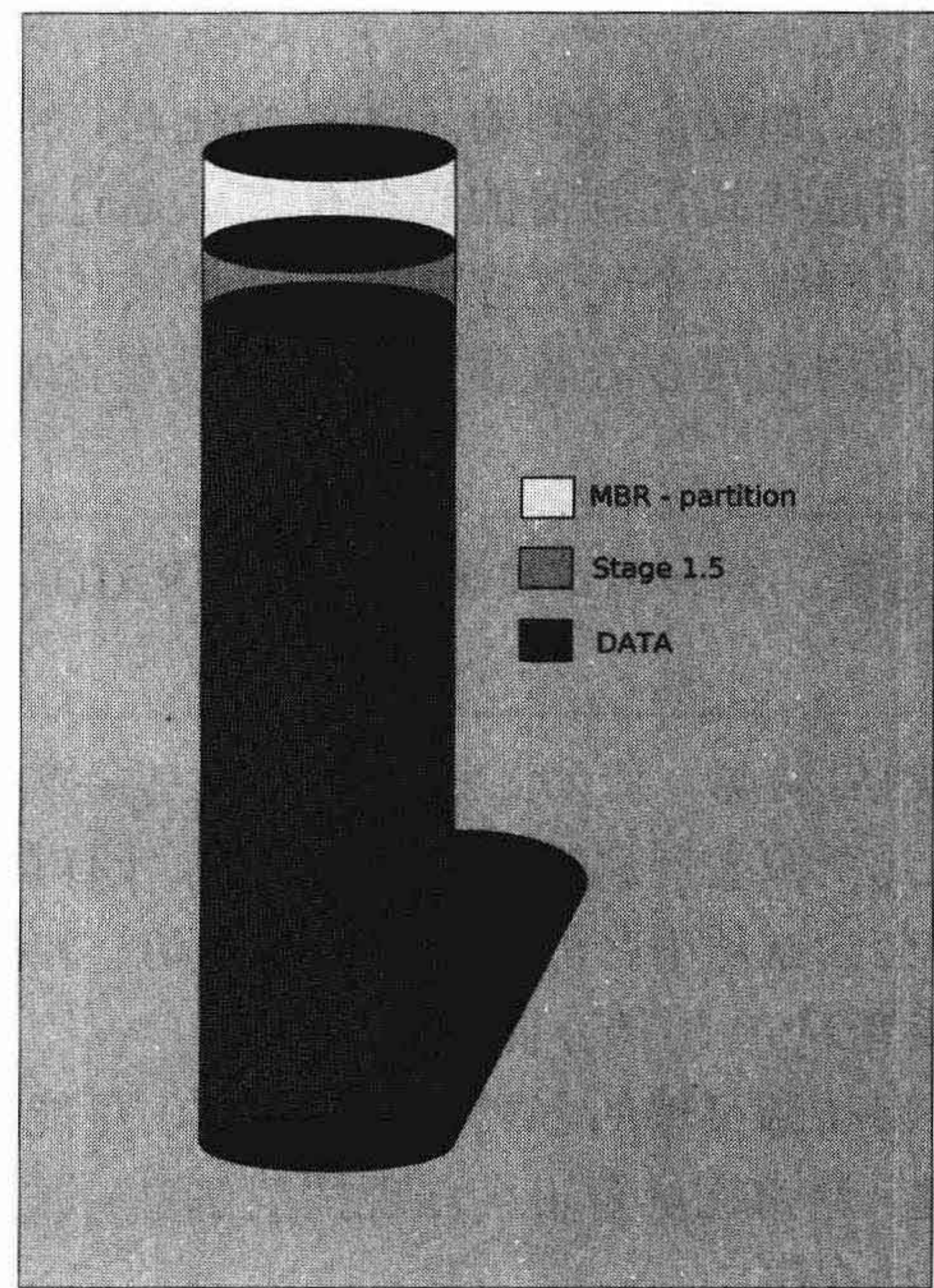
BIOS 检查一完成，BIOS 就读启动源上的 MBR 以寻找引导程序。如果它找到了引导程序，第一步就是启动它。如果它没有找到，启动过程就失败了。

GRUB 引导程序的过程有 3 个阶段。每个阶段都启动下一个阶段。BIOS 开始的第一阶段叫 Stage 1。MBR 上这个阶段的代码极少，它唯一的工作就是加载下一阶段 Stage 1.5。Stage 1.5 加载引导程序的剩余部分，即 Stage 2，并显示一个操作系统的选项菜单。

当加载 Stage 2 时，你可以看到可供选择启动进入的内核（可能是其他操作系统）菜单列表。在这里也可以与引导程序交互，给要启动的内核传递不同的命令和变量。

GRUB 在一个短暂的倒数计秒后启动默认的内核。按任意键会停止倒数计秒，并显示一个更详细的可用选项菜单以提供编辑启动配置的机会。后面“使用 GRUB 菜单”一节将详述

这些选项及其使用方法。



Partition: 分区; DATA: 数据
图 5-1 硬盘

挑选了希望启动进入的内核之后（或者等到超时后默认的内核加载时），GRUB 会立刻寻找内核的二进制代码，然后往内存加载一个叫 `initrd.img` 的特殊文件。该文件包含内核为了使用主机硬件而加载的驱动程序。

5.1.3 操作系统

加载 `initrd.img` 后，GRUB 就完成了它的任务并把控制权交给内核，后者通过初始化包括硬盘在内的硬件继续这个启动过程。它会启动操作系统，并调用一个叫作 `init` 的特殊程序。这个程序启动服务，并使主机进入工作状态。本章稍后会介绍 `init` 以及如何管理服务。

5.2 了解 GRUB 引导程序

那么现在让我们深入研究引导程序的概念和功能，并看一看它的配置方式。我们打算涉及引导程序的所有细节，因为很少有更改它的需要，而且大部分更改都自动发生，例如在安装一个新的内核软件包时。但是我们要理解它是干什么的以及它在主机启动过程的什么位置起作用。

Linux 领域存在两个主要的引导程序：LILO 和 GRUB。GRUB 引导程序是 Red Hat 和

Ubuntu 默认选项，因此这里把注意力集中于它。GRUB 代表 Grand Unified Bootloader，它是一个强大的多重启动加载器（multiboot loader）。多重启动加载器可以把主机引导到多个不同的操作系统。与 Microsoft Windows 或 Mac OS X（它们的引导工具分别为 NTLDR 和 Boot Camp）不一样，GRUB 允许在单个硬件上启动多个版本的 Linux、Microsoft Windows 和 Mac OS X。这不意味着你可以像虚拟机技术所能做到的那样同时运行所有操作系统，但是你可以通过在启动时对 GRUB 菜单的选择而分别进入这些系统。

■注：LILO 是在很多较老版本的 Linux 发行版上用作默认选项的历史遗留的引导程序。它在今天已经很少见了。有关 LILO 引导程序的信息，请参阅 <http://tldp.org/HOWTO/LILO.html>。

那么 GRUB 如何工作呢？GRUB 利用 4 项内容启动系统：一个内核文件、驱动器名、内核文件所在的分区号和一个可选的初始 RAM 磁盘。

GRUB 可以用两种方式启动。一个是直接查找并加载想要的内核，这是大多数 Linux 发行版的启动方式。GRUB 还支持一种叫做链式加载（chain loading）的启动方法；GRUB 用这种方法加载另一个引导程序，如 Microsoft Windows 的加载器，然后这个引导程序加载想要的操作系统内核。这样就使 GRUB 可以用其他操作系统的引导程序引导进入这些操作系统。

5.2.1 配置 GRUB

GRUB 引导程序是可配置的。它的配置包含在 `grub.conf` 配置文件中。这个文件在不同的发行版中位于不同位置。在 Red Hat 里，你可以在 `/boot/grub/grub.conf` 找到它（该文件通常符号链接为 `/etc/grub.conf`）。在 Ubuntu 主机里，该文件叫作 `menu.lst`，你可以在 `/boot/grub/menu.lst` 找到。

从列表 5-1 你可以看到 Red Hat Enterprise Linux 主机里一个典型的 `grub.conf` 文件。

列表 5-1 `/boot/grub/grub.conf`

```
#boot=/dev/sda
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title Red Hat Enterprise Linux Server (2.6.18-92.el5)
    root (hd0,0)
    kernel /vmlinuz-2.6.18-92.el5 ro root=/dev/VolGroup00/LogVol01 rhgb quiet
    initrd /initrd-2.6.18-92.el5.img
```

列表 5-1 显示了 GRUB 启动 Red Hat 操作系统所需的信息。这和 Ubuntu 或者其他使用 GRUB 的 Linux 主机里的 `grub.conf` 文件很相似。`grub.conf` 文件列出了每个可启动的内核（内核对应操作系统）。配置选项指定哪一个会默认加载，其他选项控制菜单的显示与配置。

让我们深入地看一看列表 5-1 中的文件。第一行 `#boot=/dev/sda` 是启动设备。它是由 Anaconda Red Hat 安装程序加载的，GRUB 会忽略它（因为所有以 `#` 开始的行都是注释，应

该忽略)。

■提示：在 Linux 配置文件中，以#符号为前缀的行一般表示注释，这样的行在处理配置文件时会被忽略。

第二行 `default=0` 指明哪一个内核（也就是操作系统）是默认加载的内核。你可以选择任何操作系统作为启动时默认加载的操作系统。如果所定义的内核不止一个，GRUB 会按从头至尾的顺序读取它们，第一个内核标号为 0，接着标号为 1，等等。例如，如果默认加载配置文件里的第 3 个内核，就要指定 `default=2`。

■注：Linux 操作系统里的计数一般从 0 而不是从 1 开始。就 GRUB 而言，一个磁盘上的第一个分区为分区 0。这种命名惯例有一个明显的例外，那就是设备名：第一个磁盘上的分区 0 为 `/dev/sda1`。

第三行是超时值，它指定 GRUB 应该等待多少秒才可以加载默认的内核。记住，如果在启动过程中按了任意键，这个计数就会中断，然后显示 GRUB 菜单。

第 4 行是闪屏图像的位置，它为 GRUB 菜单提供背景图片。它通过磁盘、分区以及在文件系统中的位置定位；这里的 `(hd0,0)/grub/splash.xpm.gz` 表示 `splash.xpm.gz` 文件位于安装在主机上的第一个硬盘驱动器 `hd0`，0 中第一个分区上的 `/grub` 目录里。

第 5 行的 `hiddenmenu` 指令告诉 GRUB 要默认隐藏它的菜单，并启动加载默认内核的倒计时。按任意键都会停止倒计时，并显示菜单。

接下来看到的是第一个内核说明。每个可启动的内核都列在这里，第一个内核是 0，第二个内核是 1，第三个内核是 2，等等（记住前面提到的默认设置）。在列表 5-1 中只有一个内核被列出。每次升级内核（一般通过安装一个新的内核软件包完成），都会另外增加一个输入项，一般同时会有一个较老的内核从列表移除并被卸载。

■提示：第 7 章将讲述软件包的安装和升级。

每个内核说明里都有许多配置变量。第一个是 `title`，它表示一条说明的开始。这也是那个内核在菜单屏幕上所显示的标题字符串。下一个选项是 `root`，它指明到哪儿寻找内核的根分区；如果它是如闪屏位置 `hd0, 0` 这样的格式，那么它就是安装在主机上的第一个硬盘驱动器的第一个分区。如果根分区定义为 `(hd1, 1)`，它就是安装在主机上的第二个硬盘驱动器的第二个分区。GRUB 在这个指定的位置寻找所希望引导的内核（GRUB 的分区表示法不同于 Linux 分区；GRUB 用 `hd0` 表示第一个磁盘，因为它使用基于 0 的表示法）。

内核选项指定要加载的内核名、它的位置以及如何加载内核，可能还给要处理的内核传递选项。在列表 5-1 中，我们将加载一个版本号为 2.6.18 的内核，它的根分区位于一个 LVM 卷组上。

■提示：有关逻辑卷管理器或 LVM 的详细信息见第 8 章。

最后，GRUB 有时需要知道一个叫 `initrd` 的特殊文件的位置。这个文件是一个微缩的文件系统，它包括使用主机硬件所需的大部分驱动程序。不是所有的内核都需要 `initrd`

文件；例如，自己编译的内核有时不需要 `initrd` 文件。比如说，如果某个内核较新的发行版本支持所需的某些新硬件或新功能，你可能就要编译它。推荐使用发行版所提供的主流内核。

■注：在 <http://grub.enbug.org/>和 <http://www.gnu.org/software/grub/manual/grub.html> 上可以找到更多有关定义内核和 GRUB 内核选项的信息。你还可以通过 `grub.conf` 的 `man` 页查找信息。

5.2.2 使用 GRUB 菜单

正如所描述的那样，主机在启动时进入默认的内核或操作系统，或者忽略它而显示 GRUB 菜单。菜单显示后，一个启动选项就会出现，你可以使用键盘上的上下箭头选择想要启动的内核。你还可以在进入内核之前编辑 GRUB 菜单，更改参数、命令和变量。

例如，你可以选择进入所谓的单用户模式或维护模式。这个特别模式在主机有损坏时使用。它限制对主机的访问只能为一个用户：通常在系统控制台。这个模式的工作方式很像 Microsoft Windows 的故障恢复控制台，它允许使用如磁盘和文件之类的资源，而不用担心冲突或者其他用户操纵主机。让我们立即进入单用户模式看看。

首先，使用键盘上的上箭头或下箭头键突出显示所希望进入的内核。按 E 键编辑突出显示的内核。你可以看到那个内核的配置，如它的位置和参数，如图 5-2 所示。

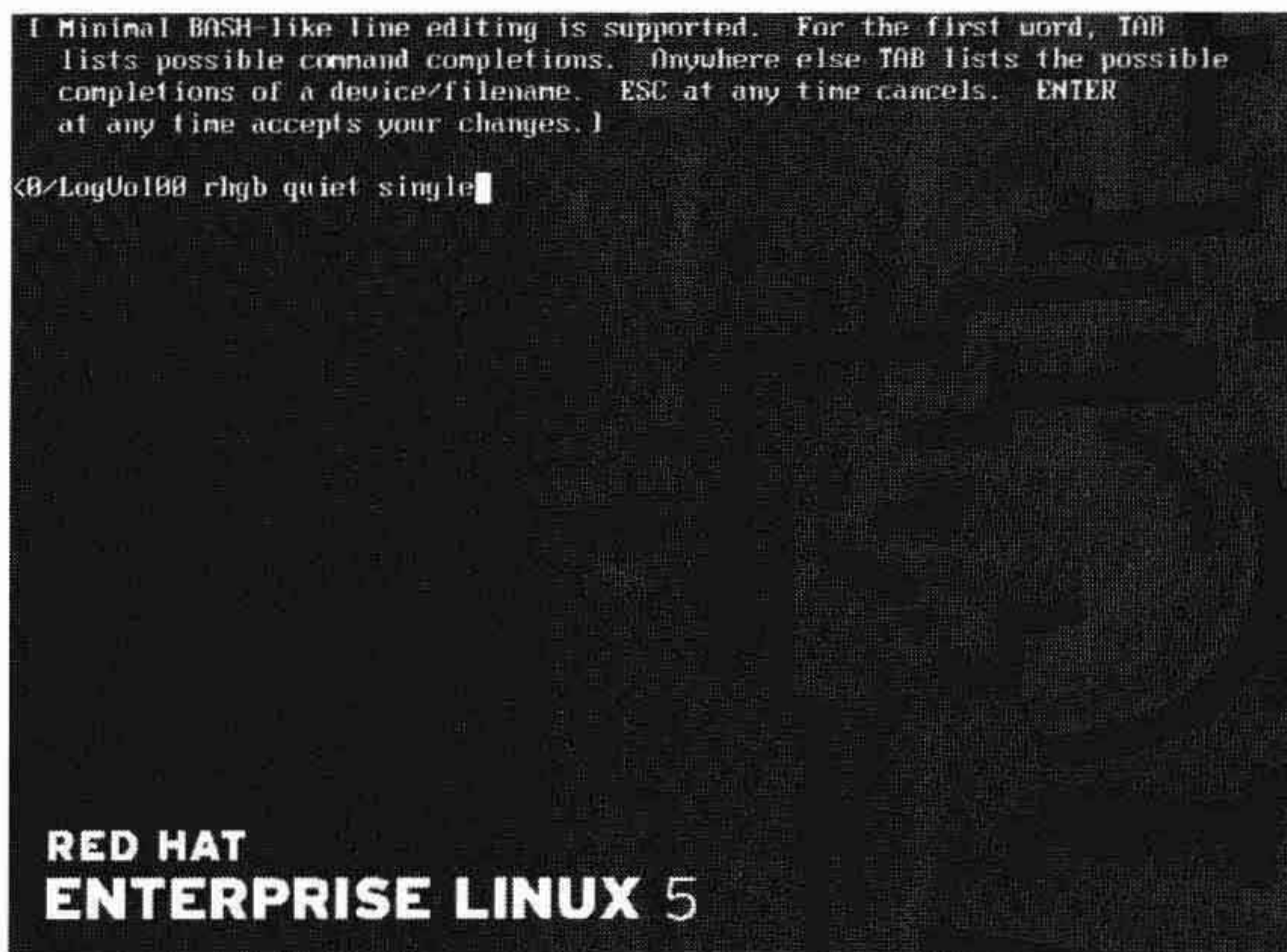


图 5-2 使用 GRUB 菜单进入单用户模式或维护模式

你可以看到第一行高亮显示，如 `root (hd0, 0)`。正如在前面概述 `grub.conf` 文件时所描述的那样，这是 GRUB 的根分区应在的位置：硬盘 0，分区 0。使用上下箭头键，高亮显示 `kernel /vmlinuz-2.x.x-x.el5 ro root` 这一行，再按 E 编辑它。

你可以看到光标已经放到了内核行的结尾，在这里指定 `rhgb` 和 `quiet` 的设置。为了进入单用户模式，在该行的结尾添加数字 1 或单词 `single`。从图 5-2 可以看到 `single` 已经添加到这一行的结尾。

如果现在启动主机，它会启动到单用户模式，然后就可以执行必要的维护工作了。不

过，对内核行的更改不长久，主机下次启动会恢复正常。

使用 GRUB 菜单几乎可以处理所有的配置设置。要查找更多的操作细节，可到 <http://www.gnu.org/software/grub/manual/grub.html> 查阅。

5.2.3 保护引导程序

这样一个易变可配置的引导程序有它的缺点。其中之一是安全性。一个不安全的引导程序在启动时可能被恶意修改。许多小办公室会把它们的服务器放到几张桌子上，而不是放在一个上锁的计算机房里，因此这些服务器对这样的攻击很敏感。强烈建议你给引导程序设置一个密码，并把它和其他密码一块存放在一个非常安全的地方。

■注：你应该永远把密码放到某个安全的地方，如保险箱或其他上锁的安全地方。有时把它放到不上网的某处（如离线备份那样）也不错，这样的话，如果站点发生了意外，你不仅可以恢复数据，而且手边有正确的密码可以访问主机。你还应该有一个所有密码的备份，并把紧急情况下可以找到这些密码的位置告诉某个信任的人。

GRUB 提供了设置引导程序密码的功能，从而对预先配置的引导过程做任何改动都需要用户输入密码。首先必须产生一个 MD5 哈希密码，然后把它添加到 `grub.conf` 文件。为此，你需使用 `grub` 命令启动 GRUB 的命令行管理器。

```
$ sudo grub
grub> md5crypt
Password: *****
Encrypted: $1$3yQFp$MEDEglsxOvuTWzWaztRly.
grub> quit
```

■提示：此密码还可以使用 `grub-md5-crypt` 命令创建。

然后，把它添加到 `grub.conf` 文件，如下所示。

```
default=1
timeout=10
splashimage=(hd0,0)/grub/splash.xpm.gz
password --md5 $1$3yQFp$MEDEglsxOvuTWzWaztRly.
```

下次启动主机时你可以注意到，如果在 GRUB 阶段中断启动过程，你就可以选择希望进入的内核或操作系统。不过，如果想编辑任何 GRUB 配置资料，你都将需要输入一个密码。要输入密码，必须按 **P** 键并输入密码。然后你就可以正常地编辑 GRUB 配置了。

5.3 启动之后发生了什么？

这样主机就找到了内核并启动它。现在开始加载操作系统，对硬件初始化，使磁盘准备就绪，分配 IP 地址，并执行其他各种任务。为了做这些事情，Linux 运行 `init` 程序，该程序

的任务就是初始化操作系统和它的服务。

■提示: `init` 程序是主机所运行的第一个进程。如果看过主机上的进程列表, 你就会注意到 `init` 进程的进程 ID 是 1。

有一种用于启动和停止 Linux 和 Unix 主机的标准方法叫作 SysV Init, 而 `init` 工具就是它的一部分。SysV 是一种定义主机在特定点所处状态的方式。SysV 通过使用运行级别 (runlevel) 的概念实现此目的。每个运行级别都包含一个应用程序和服务的列表以及一个指示是否应该开启或停止该运行级别的指示器。那么举例来说, 在主机正常启动期间, `init` 工具知道要设置什么运行级别, 然后启动在那个运行级别中所需的应用程序和服务。另一个例子发生在关机时。当你告诉主机关闭时, `init` 工具把运行级别变为 0。这个运行级别的配置是让所有应用程序和服务都停止。

SysV 有从 0 到 6 共 7 个运行级别, 每个发行版为不同的目的而使用不同的运行级别。但是有些运行级别在所有发行版中相当普遍。这包括运行级别 0、1 和 6。刚才已经介绍了用于关闭主机的运行级别 0。而运行级别 1 是单用户模式或维护模式, 它已经在本章前面介绍了。运行级别 6 在主机重启时使用。

基于 Ubuntu 的主机与基于 Red Hat 的主机上的运行级别有所不同。在 Ubuntu 上, 剩下的运行级别 2 到 5 都运行在标准的多用户模式。多用户模式是大部分发行版启动时的标准模式, 所有必需的服务通常设置为由这个运行级别启动。

相比之下, Red Hat 一般在启动时处于运行级别 5 (如果安装了 GUI 控制台) 或者运行级别 3 (对于只有命令行的安装来说)。Red Hat 具有下面这些运行级别。

- 运行级别 0: 关闭主机或使系统进入停机状态。
- 运行级别 1: 运行在单用户 (维护) 模式, 有命令控制台, 没有网络。
- 运行级别 2: 未指定。
- 运行级别 3: 运行在多用户模式, 有网络, 并启动运行级别 3 的程序。
- 运行级别 4: 未指定。
- 运行级别 5: 运行在多用户模式, 有网络和 X 窗口系统 (KDE, GNOME), 并启动运行级别 5 的程序。
- 运行级别 6: 重启主机。

5.3.1 配置 `init`

在大部分发行版中, 包括 Red Hat, `/sbin/init` 工具都是通过 `/etc/inittab` 文件配置。`inittab` 文件指定了系统应该使用的默认运行级别。它还详述了其他运行级别以及每个运行级别中要开启和停止的应用程序列表的位置。`init` 工具使用 `/etc/rc.d` 目录下一系列名为 `rc.x` 的脚本和目录, 其中 `x` 是运行级别; 例如 `/etc/rc.d/rc3.d` 目录存储了运行级别 3 中的应用程序。这个配置文件还描述了虚拟控制台 (第 3 章讲述过), 你可使用 `Ctrl+Alt+功能键` 组合访问它。

通过阅读 `inittab` 的 man 页, 你可以了解更多有关编辑 `inittab` 文件的操作。

```
$ man inittab
```


或者通过访问它的在线版本 <http://linux.die.net/man/5/inittab> 进行了解。

inittab 的 man 页说明了 inittab 文件的语法。每一行的组成是：要开启的服务的唯一 ID、服务开启所在的运行级别、要执行的动作、要运行的命令及其所有选项。相邻项之间用冒号隔开。以#开头的行都被看作注释。

```
id:runlevels:action:command -option -option
```

为了改变默认的运行级别（这通常是编辑 inittab 文件最常见的理由），你要修改 initdefault 行。这里，默认的运行级别是 5。

```
id:5:initdefault:
```

为了把默认运行级别从 5 改为 3，要把 5 替换为 3，如下所示。

```
id:3:initdefault:
```

当下次主机重启时，它将启动到运行级别 3，并开启所有与运行级别 3 相关联的程序。init 程序使用/etc/rc.d 里的一系列脚本来管理服务 and 运行级别。这些脚本列在了表 5-1 中。

表 5-1 init 控制运行级别所使用的脚本

脚本	说明
/etc/rc.d/rc.sysinit	初始化系统，并设置设备和其他任务
/etc/rc.d/rc	进入下一级目录 rcn.d 并开启或停止那里列出的 init.d 脚本
/etc/rc.d/rc.local	启动时所需的本地改动可以放在此文件中

rc.sysinit 脚本初始化像交换区这样的东西、设置其他设备以及挂载文件系统等。这使主机处于一个有效状态，然后你就可以在其上开始运行服务。千万不要在该文件中编辑并增加新任务；真要编辑增加新任务的话，可改为使用 rc.local。

init 把默认的运行级别 n 传递给/etc/rc.d/rc 脚本，然后 rc 脚本会进入/etc/rc.d/rcn.d 目录开启或停止那里的服务。这一点会在“管理服务”一节中详述。/etc/rc.d/rc.local 脚本最后执行，并且可以把希望在启动时运行的特殊命令包含进来。这些命令可能是单行命令，以完成某些不值得使用一个完整的 init 脚本说明而又希望主机启动时执行的任务。

在 Ubuntu 和一些新版的 Fedora 里，老系统 SysV Init 已经被叫作 Upstart 的新服务管理框架所替代。Upstart 仍然调用开启服务的工具 init，但它没使用/etc/inittab 文件，而是使用/etc/event.d 目录里单独的配置文件。默认的运行级别被设置在/etc/event.d 目录下一个叫作 rc-default 的文件中。本章后面“Upstart：一种新方式”一节会详述 Upstart。

提示：在 Ubuntu 里，如果想把 Upstart 更改为 SysV Init，你可以通过安装 sysvinit 软件包做到。关于软件包的安装可以到第 7 章了解。

5.3.2 在运行级别之间移动

如何不用重启就可使主机在不同的运行级别之间运行？你可以使用 telinit 或者 init 命令

在运行级别之间转换。首先，使用 `runlevel` 命令找出所在的运行级别，它会返回一个显示前一个和当前运行级别的信息，如下例所示。

```
$ sudo runlevel
N 5
```

在这里，列出的前一个运行级别是 N，当前的运行级别为 5。如果前一个运行级别为 N，那么主机先前没有运行级别（通常是因为它的运行级别自启动后就没有改变过）。

让我们假定主机启动时默认进入运行级别 5。为了把它切换到运行级别 3，你可以发出命令 `telinit 3`（`telinit` 需要以 `root` 用户身份或者通过 `sudo` 命令运行）。它会立刻处理 `/etc/rc.d/rc3.d/` 目录，并停止在这个运行级别里没有定义的程序。

```
$ sudo telinit 3
```

我们发出了 `telinit` 命令，从而把运行级别从 5 改为 3。再次发出此命令，再把 `runlevel` 从 3 改为 5。

```
$ sudo telinit 5
```

这里发生的事情是这样的。首先，`init` 进程发送转移到运行级别 3 的信号，从而进入 `/etc/rc.d/rc3.d/` 目录并终止那里没有定义的进程。同时它开启在这个运行级别定义的要开启但还没开启的进程。你可以注意到这里发生的主要事情就是 X 窗口会话停止并呈现控制台。你需要重新登入并发出 `telinit 5` 命令。`init` 工具会再次进入 `/etc/rc.d/rc5.d/` 目录，终止那里没有定义的进程，并开启 `/etc/rc.d/rc.3.d` 里定义了但还没有开启的进程。在 Red Hat 里，现在应该呈现一个 GUI X 窗口的登录界面。

5.4 管理服务

第 3 章介绍了进程和服务的概念。所运行的每个应用程序和命令都创建一个进程。有些进程随着命令完成而结束，如列出目录内容的进程。其他的进程运行时间更长，直到要求它们或者主机重启才会停止。其中一些长时间运行的进程运行一些应用程序和服务，如邮件和网络服务器、打印服务或归档服务。后台程序（`daemon`）是在后台运行的进程；就是说，它们不需要依附某个控制台。正如第 3 章所述，这样的进程都有名字；如 `sshd`、`httpd` 或者 Apache 网站服务器的 `apache`。

■注：在 Linux 上，所有进程都来源于一个父进程。进程的分叉（`forking`）使父进程生成自己的一个副本，即所谓的子进程。作为来源的父进程通常退出，并使 `init` 进程成为新的父进程。这意味着进程不需要依附某个控制台或用户会话就可以持续。如果这个父进程停止，它所有的子进程也停止。比如说，如果杀掉主机上的 `init` 进程，主机上的所有进程都会停止。这不是一个很好的主意，它可能会在主机上产生不好的后果。如果需要开启和停止所有服务，应该使用本章所讲的命令。

如前所述，运行级别用于为服务分组，从而在把主机启动到某个特定运行级别时，就可

以预期某些服务的开启和停止。当主机启动时，默认运行级别中规定要开启的服务会自动开启。你还可以在其他任何时间开启个别服务。这些服务同样可以在主机退出某个特定运行级别或者主机关闭时停止，而且还可以在任何时间手动停止。

init 程序从 inittab 文件核实它的默认运行级别后，给叫作/etc/rc.d/rc 的脚本提供运行级别的参数，并使用此脚本在运行级别之间转换。它进入适当的目录，并开启或停止那个运行级别的服务。它选择的目录和运行级别相匹配。如果默认运行级别是 3，目录就是/etc/rc.d/rc3.d。

一旦进入适当的目录，rc 脚本就停止所有前缀为 K 的服务，同时开启所有前缀为 S 的服务。服务按次序开启和停止。通过列出目录/etc/rc.d/rcn.d（这里的 n 是 0 到 6 之间的运行级别）的内容，你可以分析在每个运行级别都会开启什么服务。让我们看看列表 5-2 中 /etc/rc.d/rc3.d 目录的部分内容。

列表 5-2 /etc/rc.d/rc3.d

```
$ ls -l /etc/rc.d/rc3.d/
lrwxrwxrwx 1 root root 16 2008-04-29 06:58 K02httpd -> ../init.d/httpd
lrwxrwxrwx 1 root root 17 2008-04-29 07:31 K30postfix -> ../init.d/postfix
lrwxrwxrwx 1 root root 20 2007-11-09 04:48 K50netconsole -> ../init.d/netconsole
lrwxrwxrwx 1 root root 15 2007-11-09 04:46 K89rdisc -> ../init.d/rdisc
<snip>
lrwxrwxrwx 1 root root 19 2008-08-19 06:58 S08ip6tables -> ../init.d/ip6tables
lrwxrwxrwx 1 root root 18 2008-08-19 06:58 S08iptables -> ../init.d/iptables
lrwxrwxrwx 1 root root 17 2007-11-09 04:48 S80postfix -> ../init.d/postfix
```

可以看到，/etc/rc.d/rc3.d 目录里的所有文件都是可在/etc/rc.d/init.d/里找到的单个 init.d 脚本的符号链接（文件类型一栏中的 l 和指向引用文件的->说明了这一点）。/etc/rc.d/init.d 目录里的单个脚本包含有关如何开启、停止应用程序或服务以及恢复各自状态的指令。

所有的符号链接都以 Knumber>或者 Snumber 为前缀。这些前缀指定了在那个运行级别里服务开启和停止的优先级顺序，同时通知 init 给各自的 init.d 脚本发送一个开启（start，S）或停止（stop，K）参数。当 init 进入这些目录时，它首先停掉定义了 K 参数的那些服务，然后开启那些定义了 S 参数的服务。它开启或停止这些服务的顺序是按数字从小到大。

从列表 5-2 可以看到，Postfix 服务会在 iptables 服务之后开启，因为 S08iptables 比 S80postfix 小。（Postfix 是一个电子邮件服务器，第 10 章会介绍它；iptables 是防火墙，第 6 章会涉及它。）

这种优先级系统用来确保服务按正确的顺序开启。例如，在刚才引用的例子中，iptables 服务在 Postfix 服务之前开启。这是因为 iptables 防火墙保护主机免受网络攻击，我们希望在启动如 Postfix 这种面向网络的服务之前先启动防火墙。有关优先级的另一个常见的例子就是基于网络的服务在开启网络的服务之后开启，原因很简单，没有网络就不能运行基于网络的服务。

从列表 5-3 可以看到在停止主机时/etc/rc.d/rc0.d（运行级别 0 或停机）所做的事情。

列表 5-3 /etc/rc.d/rc0.d

```
[jsmith@au-mel-rhel-1 ~]$ ls -l /etc/rc.d/rc0.d/
total 276
lrwxrwxrwx 1 root root 17 Sep 15 03:50 K01dnsmasq -> ../init.d/dnsmasq
lrwxrwxrwx 1 root root 16 Sep 14 09:05 K01smartd -> ../init.d/smartd
```



```
lrwxrwxrwx 1 root root 22 Sep 14 09:05 K02avahi-daemon -> ../init.d/avahi-daemon
lrwxrwxrwx 1 root root 24 Sep 14 09:05 K02avahi-dnsconfd -> ../init.d/avahi-dnsconfd
lrwxrwxrwx 1 root root 16 Sep 14 09:02 K02dhcddbd -> ../init.d/dhcddbd
lrwxrwxrwx 1 root root 19 Sep 14 09:04 K02haldaemon -> ../init.d/haldaemon
lrwxrwxrwx 1 root root 24 Sep 14 09:04 K02NetworkManager -> ../init.d/NetworkManager
<snip>
lrwxrwxrwx 1 root root 18 Sep 14 09:00 K99cpuspeed -> ../init.d/cpuspeed
lrwxrwxrwx 1 root root 17 Nov 22 02:09 S00killall -> ../init.d/killall
lrwxrwxrwx 1 root root 14 Nov 22 02:09 S01halt -> ../init.d/halt
```

你的列表输出可能在内容上有所不同，因为你的主机可能安装了不同的服务。在这个列表中，每个服务都被设置为停止（K 表示停止），只有两个服务设置为开启（S 表示开启）：killall 和 halt 进程。因此，当关闭主机或者通过命令行执行 telinit 0 命令时，init 进程会进入 /etc/rc.d/rc0.d 并着手停止那些进程。

5.4.1 管理 Red Hat 上的服务

对 Red Hat 上的服务可以使用 GUI 工具和命令行管理。不过让我们首先看看 Red Hat 上的一个 init 脚本：/etc/init.d 中的 postfix 脚本。使用 head 命令查看脚本的顶端部分。

```
$ sudo head -n 5 /etc/init.d/postfix
```

它会显示/etc/init.d/postfix 文件的头 5 行，如列表 5-4 所示。

列表 5-4 RHEL 的 Postfix 脚本

```
#!/bin/bash
# postfix      Postfix Mail Transfer Agent
# chkconfig: 2345 80 30
# description: Postfix is a Mail Transport Agent, which is the program \
#               that moves mail from one machine to another.
```

在第 3 行可以看到 chkconfig: 2345 80 30。一个叫作 chkconfig 的程序利用这条信息为本章早前所描述的目录/etc/rc.d/rc2.d、/etc/rc.d/rc3.d、/etc/rc.d/rc4.d 和/etc/rc.d/rc5.d 建立符号链接。在本例中，postfix 脚本在运行级别 2、3、4、5（2345 表示这个意思）开启，运行的优先级为 80，停止的优先级为 30。chkconfig 命令为/etc/rc.d/rcn.d/目录下的/etc/init.d/postfix 脚本创建了前缀为 S80 和 K30 的符号链接。chkconfig 所使用的#description 这一行也很重要。ckconfig 和 description 两者的定义都必须给出，否则结果会出错。本章稍后“chkconfig 命令”一节将介绍如何使用 chkconfig。

postfix init 脚本余下的指令（列表 5-4 忽略的部分）用来开启、停止脚本所管理的应用程序或服务，有时还用来查询它们的状态。

补充知识：LSB 和 INIT.D 脚本

值得注意的是 RHEL 版本 5 使用前 LSB 标准写的 init.d 脚本。这些脚本已经被废弃并将用 LSB 标准改写。Ubuntu 8.04 已经在使用 LSB 兼容的 init.d 脚本。

什么是 LSB？LSB 是 Linux Standard Base 的简写，是一套各种 Linux 发行版都同意的标准。它使每个使用 Linux 的人，特别是那些在 Linux 上做开发的人，更容易做事。它试图在 Linux 配置、文件位置、软件包名字以及其他习惯方面制定共同的标准。在这里可以了解它

的更多信息: <http://www.linuxfoundation.org/en/LSB>。

至于 init.d 脚本, 需要一个新的报头语法使它兼容 LSB。在这里以 Postfix 程序为例列出一个 LSB 兼容的新 init.d 脚本。此标准规定必须在下面的关键词后面列出参数表, 有些参数由一个 \$ 前缀表示。这些 \$ 参数是 LSB 协议所描述的预留虚拟设备 (virtual facilities)。例如, \$local_fs 意思是“挂载所有的本地文件系统”。因此, Required-Start 将阻止 Postfix 的开启, 除非所有的文件系统都挂载了、日志服务在运行、域名服务器在运行、网络已连上, 而且时间已经同步过。其他的虚拟设备可以在这里找到: http://refspecs.freestandards.org/LSB_3.1.0/LSB-Core-generic/LSB-Core-generic/facilname.html。

```
### BEGIN INIT INFO
# Provides:          postfix mail-transport-agent
# Required-Start:    $local_fs $remote_fs $syslog $named $network $time
# Required-Stop:     $local_fs $remote_fs $syslog $named $network
# Should-Start:      postgresql mysql clamav-daemon postgrey spamassassin
# Should-Stop:       postgresql mysql clamav-daemon postgrey spamassassin
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: start and stop the Postfix Mail Transport Agent
# Description:       postfix is a Mail Transport agent
### END INIT INFO
```

其他的关键词也可以使用。

- Provides: 对该服务所提供的内容给出一个简短的指示。这个信息由其他服务使用。
- Required-Start: 列出此脚本开启时必须可用的服务。
- Required-Stop: 表示该服务必须在这里列出的服务停止之前停止。
- Should-Start: 规定在该服务开启之前可以开启的服务 (但不强制)。
- Should-Stop: 表示该服务应该在这里所列的服务之前停止 (但不强制)。
- Default-Start: 规定服务运行应在的默认运行级别。
- Default-Stop: 规定该服务不应该在其中运行的默认运行级别。
- Description: 提供该服务的描述。

1. 在启动和关闭时开启和停止服务

那么如何真正管理 Red Hat 主机上的这些服务呢? 打开和关闭服务的方式有两种: 一种是使用命令 chkconfig, 前面已经提到过; 另一种是使用 Red Hat 提供的 GUI 服务配置程序。让我们首先看看 GUI 服务配置程序。

GUI 服务配置

你可以用两种方式开启 GUI 服务配置程序: 在命令行里运行 system-config-services 或者在 RHEL 菜单里选择“系统”→“管理”→“服务器设置”→“服务”, 如图 5-3 所示。

这样就打开了 GUI 程序, 如图 5-4 所示。使用该程序很容易理解打开和关闭服务的过程。它通过这个窗口告知当前所在的运行级别。你可以把运行级别的配置更改为 3、4、5 或者全部。通过选择“编辑运行级别”菜单, 你可以更改要编辑的运行级别。

注意, 图 5-4 中左边面板是一个服务列表, 右边面板是一个说明和状态窗口。该图片显示“NetworkManager”是加亮的, 并给出该服务的一个简短描述及其当前状态 (停止)。

直接选定挨着某个服务名的复选框并单击“开始”、“停止”或“重启”图标, 可以执行

对服务的动作。例如，如果希望停止 httpd 后台程序，就要突出显示该服务，然后单击“停止”图标。开启和重启某个服务也是同样的过程。



图 5-3 开启 GUI 服务配置程序

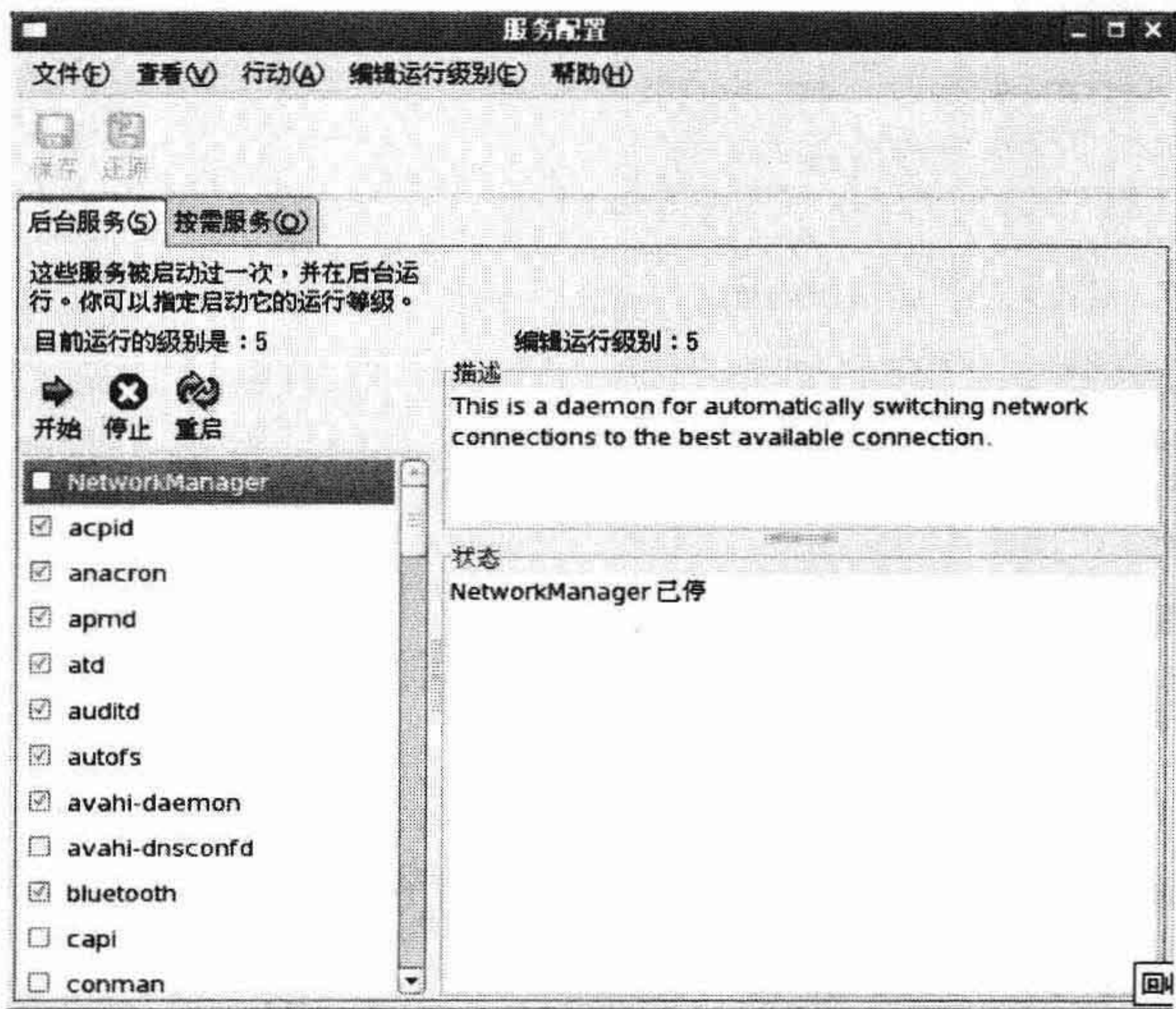


图 5-4 服务配置程序

要删除来自当前运行级别的服务，选定它，然后使用“行动”菜单删除该服务。要再添加该服务，选定它并使用“行动”菜单添加该服务。必须动手输入所希望添加的服务名。要保存改动，单击“保存”按钮，或者如果希望还原改动，单击“还原”按钮。

chkconfig 命令

Red Hat 上另一种修改服务运行级别的方式是使用 chkconfig 命令。你可以在命令行里使用 chkconfig 管理/etc/init.d 里的脚本。

如前所述，chkconfig 使用每个 init.d 脚本里的 chkconfig 报头来创建那些从/etc/rc.d 目录到/etc/rc.d/init.d/目录中特定脚本的必要符号链接。有几个选项可以传递给 chkconfig 命令，如

表 5-2 所示。

表 5-2 chkconfig 的选项	
选项	说明
--list	如果指定了某个服务，则给出该服务的相关信息。否则，列出所有的服务，并给出这些服务在每个运行级别里开启和停止的信息
--add	往 chkconfig 管理程序添加一个服务。根据 init 脚本里的信息，在每个运行级别创建一个输入项
--del	从 chkconfig 管理程序移除服务。/etc/rcn.d 目录里的符号链接被移除
--level	与服务名、想要的设置结合，管理特定运行级别的服务（例如，chkconfig --level 25 httpd off）

现在仔细分析一下这些选项。为了列出主机上的所有服务及其运行级别，你需要给 chkconfig 传递--list 选项。它会列出每个运行级别上的每个服务，并表明它是否要在那个运行级别上开启或关闭。从列表 5-5 可以看到一个例子。

列表 5-5 chkconfig --list

```
$ sudo /sbin/chkconfig --list
acpid 0:off 1:off 2:off 3:on 4:on 5:on 6:off
anacron 0:off 1:off 2:on 3:on 4:on 5:on 6:off
atd 0:off 1:off 2:off 3:on 4:on 5:on 6:off
auditd 0:off 1:off 2:on 3:on 4:on 5:on 6:off
autofs 0:off 1:off 2:off 3:on 4:on 5:on 6:off
```

正如列表 5-5 所示，如果给 chkconfig 传递了--list 选项，就会产生一个它所管理的所有服务的长列表。你还会注意到它们的名字和/etc/rc.d/init.d 目录里可找到的 init.d 文件名一样。这是因为大部分安装后台程序的软件包都会把运行 chkconfig 作为其安装过程的一部分。

如果希望根据/etc/rc.d/init.d/postfix 脚本把 Postfix 邮件服务器添加到默认的运行级别里，那么让 chkconfig 管理它，请输入下面的命令。

```
$ sudo chkconfig postfix on
```

它默认通过 chkconfig: 2345 80 30 为 init.d 脚本里指定的运行级别打开该服务。你还可以动手用 chkconfig 指定希望 Postfix 开启的确切运行级别。如下所示。

```
$ sudo chkconfig --level 35 postfix on
```

该命令会在运行级别 3 和 5 打开 Postfix。如果主机重启，它会马上在运行级别 3 或 5 开启 Postfix 服务，而不是在运行级别 2 或 4，除非明确指明。

如果希望关闭某个服务以使它在主机重启后不开启，要发出下面的命令。

```
$ sudo chkconfig postfix off
```

这样就在所有的运行级别里关闭了该服务。这意味着，当主机重启时，该服务不会在任何运行级别开启。

2. 开启和停止运行中的服务

刚刚学习了如何告诉 Red Hat 在主机启动或关闭时开启和停止服务，但是如何处理当前正在运行的服务呢？比方说，刚更新了邮件服务的版本，想重启该服务。有两种方式可完成

此事：使用 `init` 脚本和使用 `service` 命令。

使用 `init` 脚本

位于 `/etc/init.d` 目录里的所有脚本一般都可执行（就是说，它们有可执行或叫 `x` 权限的设置）。每个脚本通常还可以带下面参数中的一个：`start`、`stop`、`restart`、`reload` 和 `status`。要了解如何重启 `Postfix` 服务，请看下面的例子。

```
$ sudo /etc/init.d/postfix restart
```

它首先停止，然后开启 `Postfix` 邮件服务器。

你还可以使用下面的两个命令停止然后开启 `Apache` 的 `httpd` 服务器。

```
$ sudo /etc/init.d/httpd stop
$ sudo /etc/init.d/httpd start
```

最后，如果没有传递参数，大多数 `init` 脚本都会给出用法信息，如下所示。

```
$ sudo /etc/init.d/postfix
Usage: /etc/init.d/postfix {start|stop|restart|reload|abort|flush|
check|status|condrestart}
```

`service` 命令

基于 `Red Hat` 的发行版还有一个命令可以用于开启和停止正在运行的服务：`service`。使用 `service` 命令的好处是，它会撤除大部分环境变量并把当前工作目录设置为 `/`，以最可预测的方式开启和停止一个 `init` 脚本。下面的命令会重新载入 `Postfix`（这会重读配置文件，并且进程会尽快重启）。

```
$ sudo service postfix reload
Reloading postfix: [ OK ]
```

重新扫描配置文件后，它会自动重启 `Postfix` 服务器。当前任务一完成，所有现存的进程就重启。这样就可以在不怎么中断 `Postfix` 邮件服务器的情况下修改配置。

■提示：对于 `Red Hat` 发行版，推荐使用 `service` 命令。

5.4.2 管理 `Ubuntu` 上的服务

`Ubuntu` 和 `Red Hat` 遵循同一个 `SysV Init` 标准，但它们为达到相同的目的在执行时有些细微的差别。你已经看到，在 `Red Hat` 里开启和停止服务使用 `service` 命令，打开或关闭服务使用 `chkconfig` 或 `system-config-services`。而在 `Ubuntu` 里，要使用 `invoke-rc.d` 命令打开和关闭服务，使用 `update-rc.d` 命令管理服务。

另一方面，`Ubuntu` 与 `Red Hat` 的区别在于它采用一个较新的 `init` 机制 `Upstart`。`Upstart` 保持向后兼容性，保留了管理 `SysV Init` 脚本的能力。本章稍后“`Upstart`：一种新方式”一节将详述 `Upstart`。

1. Invoke-rc.d: 开启和停止服务

让我们先看看这些发行版的相似之处。与 Red Hat 类似, Ubuntu 主机利用/etc/rc.d 类型的目录定义在什么运行级别运行什么程序, 而单个脚本存储在/etc/init.d 里。与 Red Hat 主机不同的是, Ubuntu 没有/etc/inittab 文件来管理默认的运行级别, 而且那些决定在每个运行级别中要开启(和停止)什么服务的符号链接所在的目录位于/etc。Ubuntu 主机不使用 service 命令开启和停止服务, 也不使用 chkconfig 管理 init 脚本。

■注: Ubuntu 后来的版本, 从 8.10 往后, 都有 service 和 chkconfig 命令可用(在叫作 chkconfig 的软件包里)。

在 Ubuntu 8.10 版本之前的主机里(像我们的 Ubuntu 服务器版本 8.04), 要开启和停止服务, 就要运行 invoke-rc.d 命令。而在 Red Hat 里, 还可以执行单个脚本。所有的 init 脚本都有 start 或 stop 选项, 此外许多脚本还接受 restart、reload 和 status 选项。如果想开启 Postfix 程序, 就要发出下面的命令。

```
$ sudo invoke-rc.d postfix start
```

如果接下来想停止 Postfix 服务, 就要发起下面的命令。

```
$ sudo invoke-rc.d postfix stop
```

或者执行单独的脚本:

```
$ sudo /etc/init.d/postfix stop
```

■注: 当然了, 你可以自行阅读这些脚本, 看看它们是做什么的以及它们还接受其他什么参数。

2. Ubuntu: 打开和关闭服务

在 Ubuntu 里有一个叫作 sysv-rc-conf 的基于文本的 GUI 工具可用来管理主机的服务。sysv-rc-conf 程序与 Red Hat 里的 system-config-services 相同。利用它可轻松查看特定运行级别中设置要运行的程序, 并且可以切换它们下次重启时的开和关的状态。这个命令默认不安装, 因此要先安装它。

```
$ sudo aptitude install sysv-rc-conf
```

■注: 第 7 章将深入研究软件包管理和软件安装的方式。

sysv-rc-conf 工具是一个基于文本的 GUI 工具, 但同样可以在命令行里通过给它传递一些参数来使用它。它的语法非常类似于 Red Hat 里的 chkconfig 命令。

在命令行里, 通过发出下面的命令可以列出主机上的服务。

```
$ sudo sysv-rc-conf-list
```

此命令的输出类似于 Red Hat 上 chkconfig --list 命令的输出。你还可以通过命令行以与 chkconfig 相同的方式关闭服务。


```
$ sudo sysv-rc-conf --level 35 postfix on
$ sudo sysv-rc-conf --level 35 postfix off
$ sudo sysv-rc-conf postfix on
$ sudo sysv-rc-conf postfix off
```

sysv-rc-conf GUI 基于文本，因此快速而易用。从图 5-5 可以看到它的样子。

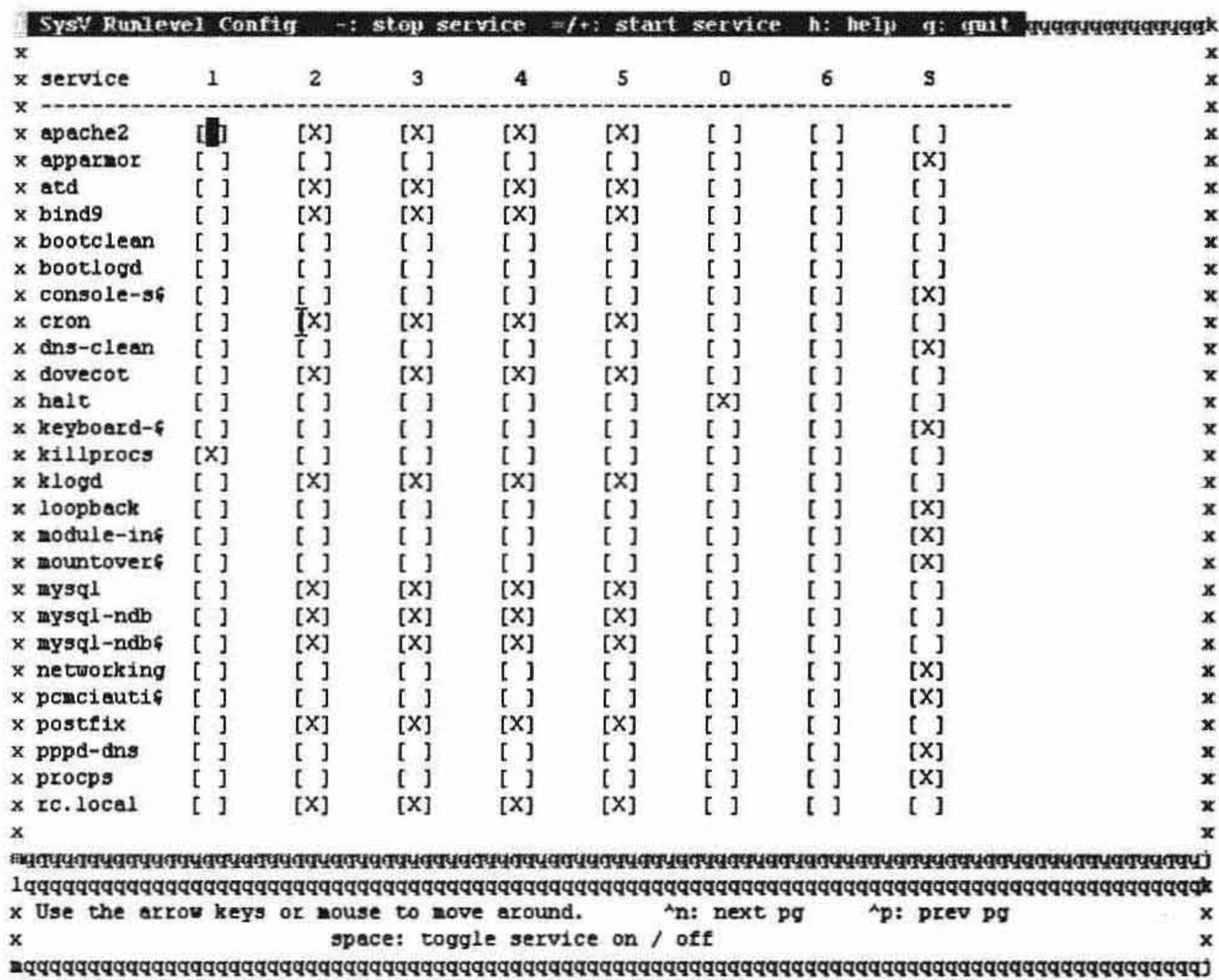


图 5-5 Ubuntu 上管理服务的 sysv-rc-conf 程序

要打开或关闭服务，只需使用箭头键把光标移动到想要管理的服务上，然后使用空格键切换 X 的开关状态。打开或关闭 X 取决于是否希望该服务在那个运行级别运行。按 Q 键保存并退出。

Update-rc.d：在命令行上打开和关闭服务

在 Ubuntu 的命令行里管理服务通过 update-rc.d 命令来执行。类似于 Red Hat 主机上的 chkconfig，update-rc.d 可以创建/etc/rc?.d 目录的符号链接或者按指示移除它们。update-rc.d 命令可带表 5-3 所列的选项。

表 5-3 update-rc.d 的选项

选项	说明
start	允许明确声明运行级别和启动顺序
stop	允许明确声明想要停止的服务次序和运行级别
defaults	update-rc.d 脚本会用默认的开启次序（S20）在运行级别 2、3、4 和 5 中创建启动符号链接，用默认的停止次序（K80）在运行级别 0、1 和 6 中创建停止符号链接
remove	只要文件/etc/init.d/script-name 已经移除，就从每个运行级别里移除符号链接（更多信息见-f）
-n	在什么都不改变的情况下演习将要发生的事情
-f	在与 remove 选项一起使用时，强制从/etc/rcn.d 目录移除符号链接，即使 /etc/init.d/script-name 仍然存在

正如较早前所看到的那样，在 Red Hat 主机上，postfix init.d 脚本中提供了 chkconfig 信息，而在 Ubuntu 主机上不是这样，update-rc.d 不需要 init.d 脚本提供任何特定信息，它直接把 init.d 脚本链接到/etc/rcn.d 目录，而且通常使用默认设置：运行级别 2、3、4、5 与开启优先级 S20；运行级别 0、1、6 与停止优先级 K80。

然后，你可以在命令行里发出 update-rc.d 命令来处理在特定运行级别中运行的服务。例如，为了利用 Ubuntu 的默认选项打开一个服务，可以发出下面的命令。

```
$ sudo update-rc.d postfix start defaults
```

如前所述，在默认设置的情况下，init.d 脚本被符号链接到/etc/rcn.d 目录并得到标准的开启和停止优先级：20 和 80。

使用下面的命令可以指定希望服务开启的运行级别及其优先级。

```
$ sudo update-rc.d postfix start 23 40
```

这里设置该服务在运行级别 2 和 3 开启，优先级为 40。

要在运行级别 2 关闭该服务，发出下面的命令。

```
$ sudo update-rc.d postfix stop 2
```

上述命令会把一个 K80postfix 符号链接添加到/etc/rc2.d 目录里。

为了从所有的运行级别中移除服务，要发出下面的命令。

```
$ sudo update-rc.d postfix remove
```

如果正在试图移除的服务其 init.d 脚本仍然存在于/etc/init.d 目录，这种情况会在 Postfix 没有卸载时发生，那么除非使用了 -f 选项，否则就会得到一个错误信息。在这种情况下，要发出下面的命令。

```
$ sudo update-rc.d -f postfix remove
```

本章前面的补充知识“LSB 和 init.d 脚本”中提到了 LSB 项目，现在你可能开始理解为什么此项目是使 Linux 更加用户友好的重要一步了。Red Hat 和 Ubuntu 二者为达到同样的结果而采用了不同的方式，因此标准化工作很受欢迎。现在让我们看一看未来 Linux 主机上的服务管理方式。

5.4.3 Upstart：一种新方式

正如早前指出的那样，Ubuntu 没使用 SysV 来启动它的主机。较新版本的 Ubuntu（自版本 6.10 往后）和版本 9 之后的 Fedora 使用了一个叫作 Upstart 的新标准。Upstart 背后的想法是要创建一个可在 Linux 主机上开启、停止、监视服务以及回应服务事件的全面的 init 进程。它是对 SysV Init 范式的改写和扩展；设计者创建 Upstart 并不是完全替代 SysV Init，而是为了仿效 SysV Init 直到所有人彻底转换过来。

前面提到 SysV Init 脚本按照它们在相应/etc/rc.d 目录里的顺序由 init 命令依次运行。SysV Init 根据运行级别决定主机启动时运行什么服务。在关机之前它对此服务不感兴趣。相比而言，Upstart 为了后向兼容只保留了运行级别的概念，并使用事件来决定主机运行什么服务，从而显示对 init 的不同的关注点。

Upstart 下的新 init 进程基于事件的后台程序，它使用事件触发进程的开启或关闭。一个事件是一个可告知 init 的状态改变。事件可能是添加一个外围设备，比如插入 U 盘。然后内核可能发送一个事件通知告诉 init 这个动作。这个事件反过来可能触发其他任务的启动或停止，这取决于 init 控制范围内的任务定义。

Upstart 在它的命令 jobs 下集中调用进程，可以使用 initctl 命令和这些任务交互。任务可能是服务（service）或作业（task）。服务是持久的，如邮件服务器，而作业执行一个功能后退回到等待状态，如备份程序。在/etc/event.d 目录下可以找到任务的定义文件（或 Upstart 脚本）。

列表 5-6 是 Upstart 里 logd 后台程序的一个所谓的任务定义（service job definition）。在本例中，logd 是一个事件记录后台程序，需要把它的输出写入控制台。作为事件记录后台程序，重要的是，如果它由于某种原因停止了，最好让它尝试重启自身。在这里让我们浏览一遍这些要点。

列表 5-6 /etc/event.d/logd

```
# logd
# This service is started automatically by init so that the output from
# other services can be logged.
description "service logging daemon"
author "Scott James Remnant <scott@ubuntu.com>"
stop on runlevel 0
stop on runlevel 1
stop on runlevel 6
console output
exec /sbin/logd
respawn
```

配置选项 stop on 是一个事件定义，它告诉 Upstart 在检测到运行级别 0 事件时停止 logd。该选项在运行级别 1 和 6 的情况类似。大家知道，运行级别 0、1、6 是特殊的运行级别，它们或者关机、重启主机，或者置主机于维护模式。console output 选项把 logd 的输出导向控制台。exec 选项把 Upstart 导向运行 logd 后台程序要执行的二进制代码。最后，respawn 选项指示 Upstart 在某个任务意外停止时重启它。

虽然 Upstart 不同于 SysV，没有运行级别的自然概念，但它必须能够向后兼容 SysV Init 脚本。因此，在/etc/event.d 里可以看到多个 rc? 定义文件。这些文件有助于执行/etc/init.d/rcn 脚本以效仿老版的 SysV 脚本。然后它就会运行到老的/etc/rcn.d 目录，并开启和停止那个特定运行级别下的服务。

因为使用了老的 rcn.d 目录，所以较早讲过的服务管理工具在使用 Upstart 的系统里仍然工作的很好。

5.5 关闭和重启主机

在 GUI 和命令行里都有好几种关闭和重启 Linux 主机的方式。从命令行关闭主机，可以使用名字很相称的 shutdown 命令。


```
$ sudo shutdown -h now
```

这使主机开始关闭并立即进入停机状态（本例中指定了-h 选项）。后台发生的事情是 shutdown 程序告诉 init 把运行级别变为 0。它会立即着手杀掉所有服务，然后运行两个特别的程序 killall 和 halt，前者杀掉没有正常关闭的剩余服务，后者卸载文件系统并关掉主机的电源。

你还可以使用 shutdown 命令设置主机重启，不过这次给出重启选项-r。

```
$ sudo shutdown -r -t: 5 "This host is being rebooted for maintenance"
```

■注：如果误用了 shutdown，通常可以用 Ctrl + C 键取消预定的关机任务。

可以看到，这里传递了-r 参数来重启主机。还通过传递-t: 5 参数告诉它要有 5 分钟的宽限时间。这个时间可以按照需要随意指定。最后还给出了一个警告信息，略述了主机重启的原因。这条信息会发送给当前正在登录的所有用户。

5min 之后，shutdown 程序告诉 init 把运行级别变为 6，在运行级别 6 里，除了运行 reboot 程序替代 halt 程序之外，它执行与在运行级别 0 中一样的动作。

从 GUI 关闭主机很容易。单击“系统”菜单，然后选择“关闭”选项。

5.6 使用定时任务调度服务与命令

本章要介绍的最后一种服务管理类型是调度。你可能已经熟悉 Microsoft 的任务计划，任务计划可只运行一次，或者按照给定的分钟、小时、天、周或月定期重复运行。在 Linux 里对等的概念叫 crontab（chronograph table 的简写）。它的目的是按照主机的时钟在设定的时间提交任务。任务可以是所需要的任何脚本或应用程序。在 crontab 里通常可以找到维护类型的任务。这些任务可以设置为每晚、每周或每月运行某类脚本，比如删除/var/log/httpd 目录里所有超过 2 个月的文件。

定时任务（crontab 所执行的任务）在/etc/crontab 文件所规定目录下的一系列脚本中指定。这些任务被称为系统定时任务（system cron job）。/etc/crontab 文件里的目录列表看起来如下所示。

```
$ less /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

不要编辑这个文件，因为作为系统的 crontab 文件，它很可能在 crontab 每次更新时被替

换为新版本。这意味着任何改动都会丢失。而且如果编辑时犯错，还可能导致其他问题。无论如何，它的确提供了一个有关 `crontab` 文件语法的很好的例子。

提示：如果单个主机用来运行多个虚拟服务器，应该修改每个虚拟服务器定时任务开始的时间，以确保它们不会全部同时运行。多个虚拟机全部同时运行它们的日常定时任务会影响系统性能。

该文件顶端列出的是 `SHELL`、`PATH`、`MAILTO` 和 `HOME` 环境变量（第 4 章讲过）。以 `#` 开头的行是注释，可以忽略。再往下可以看到数字或 `*` 号填充的 5 栏、`root` 栏、`run-parts` 栏和末尾的目录列表栏。

头 5 栏代表分钟、小时、日期、月份和星期。让我们看看最后一行。

```
42 4 1 * * root run-parts /etc/cron.monthly
```

在这里，42 是指第 42min，小时数是 4（时钟基于 24 小时制），然后 1 代表月份的第 1 天。因此 `crontab` 在每月 1 日凌晨 04:42 运行最后一行。

注：对于月份、日期和星期这些栏，你还可以指定标准的三字母缩写，例如 `sun` 代表星期日，`aug` 代表 8 月。

带 `*` 的列表示所有值都有效，运行时不受限制。还以最后一行举例。

```
42 4 1 * 3 root run-parts /etc/cron.monthly
```

这里把星期这一栏修改为 3。那么预定任务会在每月 1 日凌晨 04:42 运行，而且每周周三也运行。

注：星期的开始为星期日，用 0 表示，这样星期六就用 6 表示。如果在日期这一栏指定一个值，那么日期是累计的。因此在日期那一栏列出的所有日子和星期那一栏列出的每个工作日都会执行某项任务。

你还可以自动指定重复执行的任务，如下所示。

```
*/2 4 1 * 3 root run-parts /etc/cron.monthly
```

这里没有指定一个确切的执行时间，而是用了符号 `*/2`。这个符号告诉定时任务调度程序每当分钟数被 2 整除时都要运行此任务。利用它可以做一些强大的事务。例如，你可以在小时这一栏使用 `*/4`，那么该任务会在每 3 个小时后的 1 个小时中每隔 1 分钟运行一次，如下所示。

```
*/2 */4 1 * * root run-parts /etc/cron.monthly
```

逗号用来表示一个值的列表，如下所示。

```
2 0,1,2,3,4 1 * * root run-parts /etc/cron.monthly
```

这样该任务就会在上午的 12:02、1:00、2:00、3:00 和 4:00 运行。

还可以指定数字的范围，如下所示。

```
2 0-4,12-16 1 * * root run-parts /etc/cron.monthly
```

在这里，这个命令会在上午 12 点到 4 点和下午 12 点到 4 点这两个时段中每个小时的第

2min 运行。

接下来的 **root** 一栏表示运行该程序所需的用户身份。当添加自己的定时任务（或脚本）时，你可以把它设置为任何有效的用户。

run-parts 选项是要运行的命令。**run-parts** 是一个运行指定目录内所有可执行脚本的特别命令。在本例中，**run-parts** 会把目录更改到 **/etc/cron.hourly**、**/etc/cron.daily** 等，并运行在那里找到的可执行文件。

让我们查看其中一个系统定时任务目录，以 **/etc/cron.daily** 目录为例，并分析 Red Hat 主机上已有的脚本之一。这些是系统定时任务，如果没有另外规定，它们会一天运行一次。

■注：如果需要的话，你可以编辑 **crontab** 目录中的脚本；不过，任何改动都可能在软件包更新时被覆盖。你还可以在这些目录里添加自己的脚本，以使它们每小时、每天、每周或每月运行。

```
$ ls -l /etc/cron.daily/
-rwxr-xr-x 1 root root 379 Dec 19 2006 0anacron
-rwxr-xr-x 1 root root 118 Oct 1 00:06 cups
-rwxr-xr-x 1 root root 180 Oct 22 2007 logrotate
-rwxr-xr-x 1 root root 114 Jan 16 2008 rpm
-rwxr-xr-x 1 root root 290 Nov 26 2006 tmpwatch
```

让我们看其中一个文件 **/etc/cron.daily/rpm**，看看它里面是什么。

```
$ less /etc/cron.daily/rpm
#!/bin/sh

/bin/rpm -qa --qf '%{name}-%{version}-%{ release}.%{arch}.rpm\n' 2>&1 \
    | /bin/sort > /var/log/rpmpkgs
```

这个每天都执行的脚本对主机上的 **RPM** 软件包排序，并组建成 **/var/log/rpmpkgs** 文件。

■注：第7章将深入讲解 **RPM** 包。

单个用户也可以创建 **crontab**。创建和编辑现有的 **crontabs** 要使用 **crontab -e** 命令。如果你的 **crontab** 还不存在，此命令会在 **/var/spool/cron/username** 里创建一个 **crontab** 文件。

用户的定时任务语法和早先看到的系统的 **crontab** 文件基本相同，只有一个差别。在系统 **crontab** 文件里你只能指定用户字段。让我们看看用户 **jsmith** 使用 **-l** 或者叫列表选项创建的一个 **crontab** 例子。

```
$ crontab -l
*/2 * * * * [ -e /tmp/log ] && rm -f /tmp/log
```

你可以看到该用户制定的所有定时任务的列表。这是一连串简单命令。它首先检查 **/tmp/log** 文件是否存在，如果存在，就移除它。时间设置为每 2min（*/2）运行一次。

作为一个特权用户，你可以通过发出带 **-u username** 选项的 **crontab** 命令查看别人的定时任务。

```
$ sudo crontab -u ataylor -l
1 2 * * * /usr/local/bin/changeLog.sh
```

你还可以通过发出同样的 **crontab** 命令和 **-u username -e** 选项编辑其他人的定时任务。


```
$ sudo crontab -u ataylor -e
```

这样就可以编辑该用户的 `crontab`。

你还可以通过发出带 `-r` 选项的 `crontab` 命令移除你的 `crontab` 或其他用户的 `crontab`。

```
$ sudo crontab -u ataylor -r
```

这样就移除了 `ataylor` 的 `crontab` 文件：`/var/spool/cron/ataylor`。

主机里有一个服务监视定时任务以及所有对它的改动。它还执行所安排的单独任务。这个服务叫作 `crond`，你可用 Red Hat 里的 `service` 命令或 Ubuntu 里的 `invoke-rc.d` 命令通过 `init.d` 脚本开启和停止此服务。

```
$ sudo service crond start/stop/reload
```

5.7 小 结

本章探究了主机的启动过程及其背后的进程，如 `init` 后台程序。你学习了如何管理服务、如何开启和停止它，以及如何添加和删除来自不同运行级别的服务，还了解了 LSB 项目，对新的 Upstart `init` 后台程序有了一个大概的了解。

你现在应该能够做下面的事情。

- 描述 Linux 的启动过程。
- 使用、配置和保护 GRUB。
- 描述 `init` 脚本，包括使用 LSB 标准的脚本。
- 在 Red Hat 和 Ubuntu 上开启和停止服务。
- 打开和关闭特定运行级别的服务。
- 用 `crontab` 安排任务。

下一章将介绍如何配置网络，探讨防火墙，并介绍 Linux 的安全性。

第 6 章 网络与防火墙



迄今为止，我们已经阐述了 Linux 的一些基本功能，但是其中最关键的是网络。正是通过网络主机才可以与其他主机对话，应用程序才可以与用户、与世界通信。本章将讲述如何设置主机的网络以及如何使用防火墙保护网络免受攻击。

■注：防火墙是控制通过网络对主机访问的一连串的规则。

本章将介绍如何配置网卡或网络接口以及如何给它们提供 IP 地址。你会学到如何连接到其他网络以及如何解决连接故障。

本章还将介绍一个叫作 Netfilter 的软件程序，它是所有的 Linux 发行版共用的防火墙。你会学到如何管理防火墙以及如何编写防火墙规则。为了做到这一点，将介绍 Netfilter 的管理界面 iptables。最后还会阐述如何能够使用 TCP Wrappers 保护主机上正在运行的后台程序。

介绍了网络配置的基础知识之后，本章还将阐述如何配置一个可能正适合你的环境的范例网络。到本章结束，你应该掌握为你的环境配置合适网络的技巧。

本章会用到网络术语。我们不期望你是网络专家，但是假定你有一些基本的知识。如果你感觉知道的不够，推荐看一下这些站点和教程。

- <http://handsonhowto.com/2007/lan101/>
- <http://www.w3schools.com/tcpip/default.asp>
- <http://compnetworking.about.com/od/workingwithipaddresses/l/blip.htm>
- <http://en.wikipedia.org/wiki/TCP/IP>

■注：iptables 用来保护网络服务。在第 9 章你会学到更多有关如何在主机上运行如 DNS 和 DHCP 这种服务的知识。

6.1 网络与连网概论

网络由硬件和软件两者构成。网络根据其大小和所需互联层次的不同而有不同的复杂度。

小公司可能有一个简单的网络。也许有一个网络服务器和邮件服务器，可能还有一个文件/打印服务器（有时所有这些服务器实际上是一台主机）。毫无疑问，你要有一个 Internet 连接，可能还想与组织中的其他人共享此连接。

企业和所做工作的性质很大程度上决定了如何建立网络。企业在起步阶段通常只有一台主服务器，它几乎要实现这个企业所需的所有功能。它可能是 DHCP、DNS、文件、邮件和网站服务器全放到一个主机里。那些熟悉 Microsoft 产品的人可以把它看作与 Windows 小型商业服务器类似的产品。但是随着业务的增长，这些组合在一起的功能中的一部分开始移到它自己的主机上。较大的企业很少会把整个公司的 IT 基础架构都放在一台有那么多角色的主机里。在可能发生单点故障的地方应该避免这种情况的发生，但是小公司很少能奢侈到为他所希望提供的每个服务都提供一台主机。

■注：如果你的企业确实有单点故障，比如很多服务在一台主机上，那么备份和恢复就变得很关键。数据丢失对于你的企业来说是个灾难，因此必须具备备份和恢复主机及数据的能力。有关如何实现备份和恢复策略的详情，请参阅第 13 章。

接下来需要硬件互连部件。如果一个办公室里的连网用户要连接到单个网络，那么就需要网线、接线板开关，可能还需要一个可创建无线网络的无线接入点。

■注：无线网络是一种好用又便宜的网络扩展方式。它不需要昂贵的网线和交换机，并且职员在办公室里的位置可以更灵活。尽管如此，这还是提出了一些挑战。如果保护不当，攻击者会利用无线网络连接并监听你的网络，并且无线网络的性能没有有线网络（带物理网线的网络）那么好。例如，对于传输大量的数据，速度较快的还是有线的连接而不是无线连接。如果正考虑使用无线网络，推荐阅读 http://en.wikipedia.org/wiki/Wireless_security、http://www.practicallynetworked.com/support/wireless_secure.htm 和 http://www.us-cert.gov/reading_room/Wireless-Security.pdf 上的信息。

我们将先说明如何在单个主机上配置网络，并介绍配置更广网络所需的工具和命令。

为了说明如何配置网络，我们将使用已创建的一个范例网络。从图 6-1 可以看到这个范例网络。

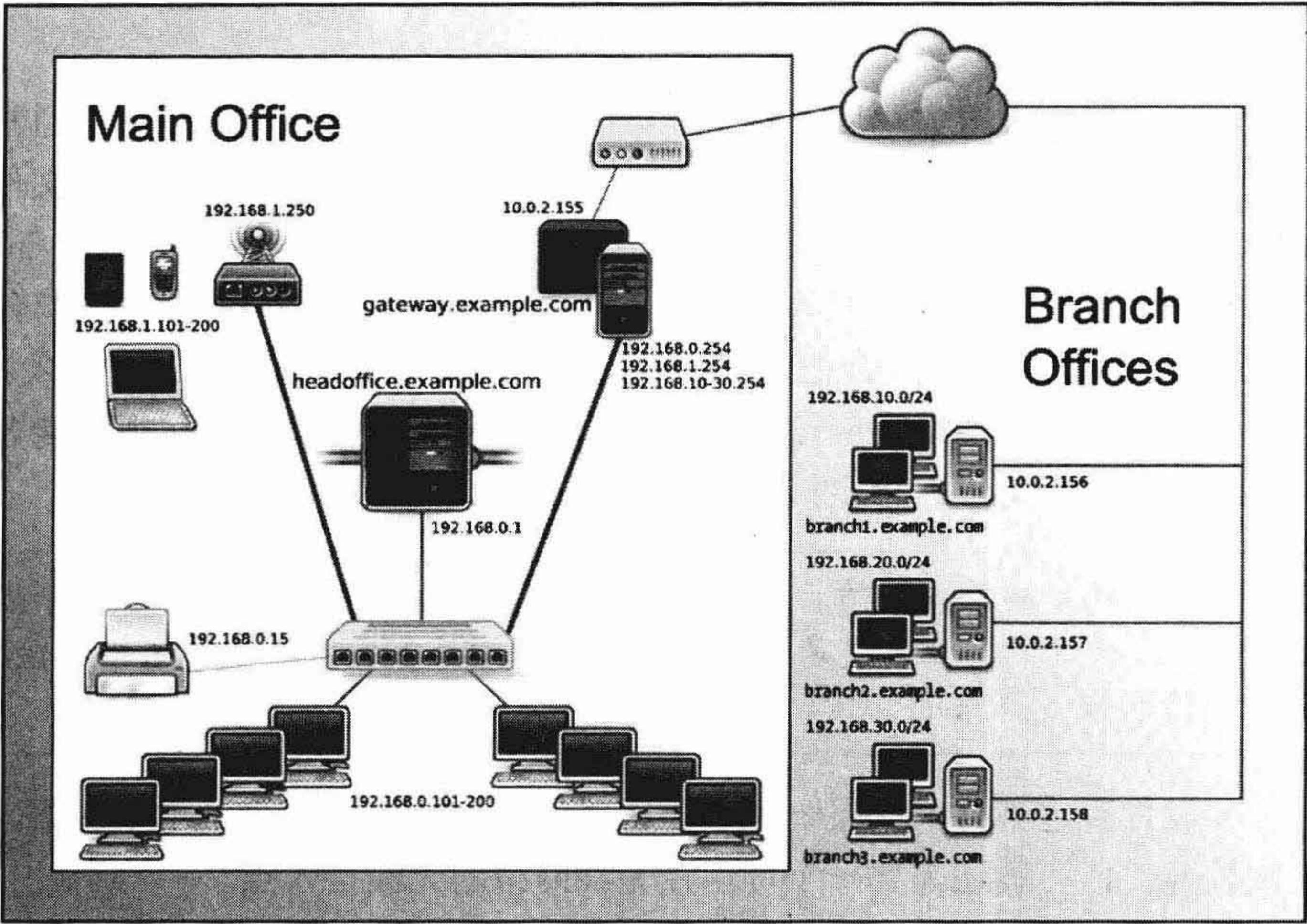
这是完整网络示意图。到本章结束，这个图将不再像现在这样令人畏缩。我们将介绍它的组件及其配置方式，还会阐述所有 IP 地址的用途，以及如何能够穿越和阻止穿越网络。

本章将说明如何配置范例网络的元件。我们会配置一个防火墙/路由器主机，并称它为 `gateway.example.com`。它有多个 IP 地址，每个以它作为路由器的网络都对应一个 IP 地址，在内部网络中是 192.168.0.254，在无线网络中是 192.168.1.254，还有一个外部 IP 地址是 10.0.2.155。

我们还会配置一个主服务器，并称它为 `headoffice.example.com`。它在内部网的 IP 地址为 192.168.0.1。它会路由到其他网络，比如无线网络、分支办公室以及 Internet（通过 `gateway.example.com` 主机）。

可以看到，我们已经把网络划分为独立的部分，并选择了不同的网络地址来显示。正如所提到的那样，无线网络的网络地址是 192.168.1.0/24，而提供这种方便的无线接入点的 IP

地址为 192.168.1.250。



Main Office: 主办公室; Branch Offices: 分支办公室
图 6-1 我们的范例网络

我们的分支办公室有一个单独的网络地址，范围从 192.168.10.0/24 到 192.168.30.0/24。这样，每个分支办公室都可以有 254 个节点（或设备），如果需要的话，还可以扩展。

我们还有一个本地的有线网络，网络地址为 192.168.0.0/24。主服务器 headoffice.example.com 的 IP 地址为 192.168.0.1，它可以通过将要建立的从那台主机到那些远程分支主机的 VPN 网络（第 14 章将介绍 VPN）与分支办公室通信。

本地有线网络的台式机有一个地址池可以使用，范围是 192.168.0.101~192.168.0.200。它可容纳 100 个节点，并且在需要时可以扩展。

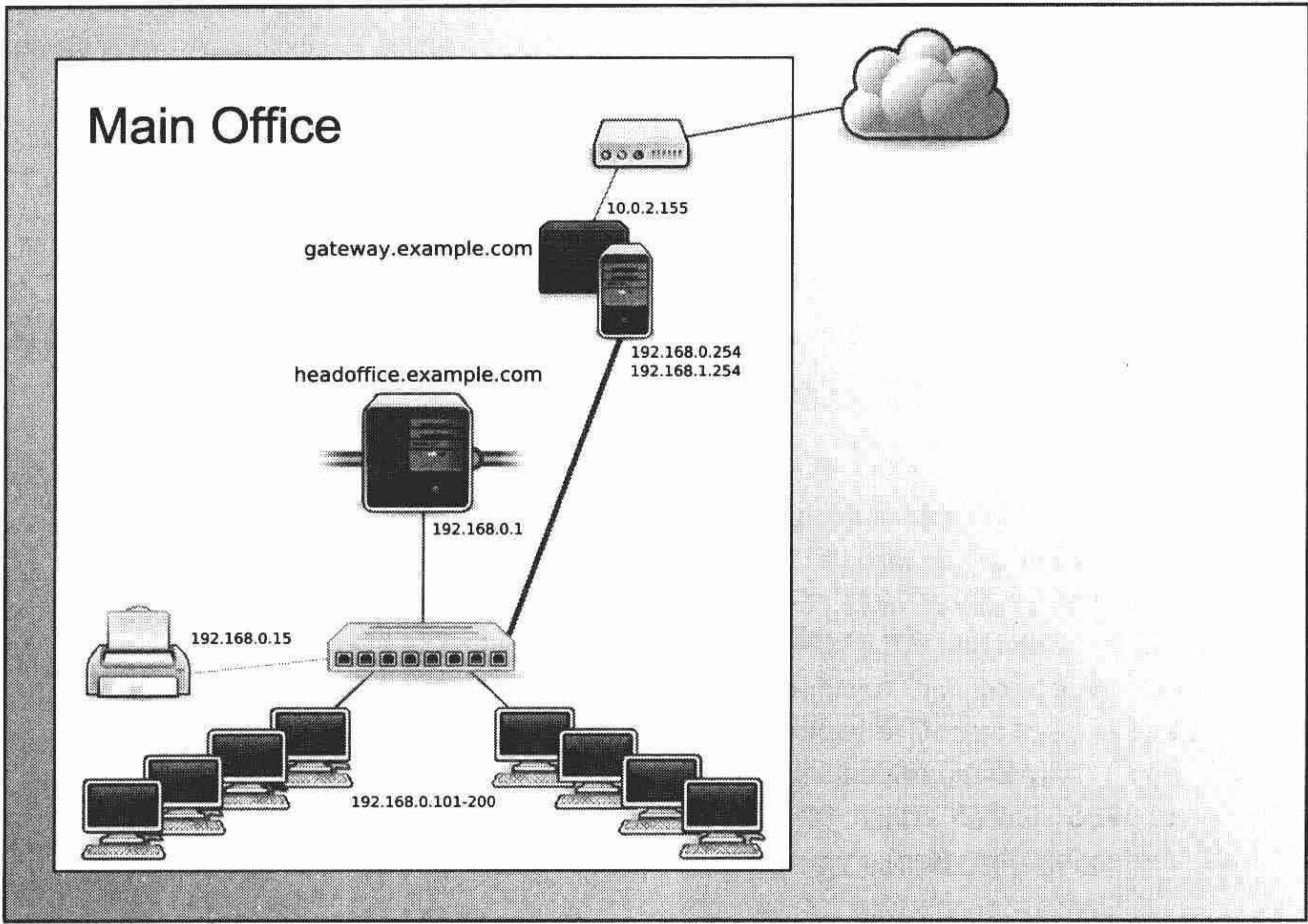
实际上，对于如此规模的网络，也许需要更多服务器，并且这些服务器分散放置在分支办公室里。不过，考虑到本章的目的，我们将专注于图 6-2 所给出的图景，从中可以看到网络被分解为更小的模块。

在这里，我们可以专注构建办公室的首要服务器 gateway.example.com 和 headoffice.example.com。

我们将说明如何在 gateway.example.com 主机上设置一个 PPPoE 连接以使之担当连接到 ISP 和 Internet 的防火墙/路由器主机。

■注：PPPoE 即 Point-to-Point Protocol over Ethernet，它是一种用来从 ADSL 调制解调器连接到 Internet 的方法。

主要的主机 headoffice.example.com 将向公众提供邮件、网络和 DNS 服务（这些内容将在第 9 章、第 10 章和第 11 章介绍）。



Main Office: 主办公室
图 6-2 本地有线网络

使用 gateway.example.com 防火墙主机接收和路由来自 Internet 并通过内部网络到达主要主机的流量。headoffice 主机也会为本地网络提供 DNS、DHCP、NTP、SMTP、IMAP、HTTP 和 HTTPS 服务。我们还会说明如何在网络中把流量从一台主机路由到另一台主机。先从接口的设置开始。

6.1.1 从接口开始

将要介绍的第一个连网元件是接口或网络接口。主机上每一个网卡都是一个网络接口。在第 2 章说明如何安装一个发行版以及配置一个网络时，其实就是在配置一个接口。

■提示：接口一般处于两种主要的状态——up 和 down。处于 up 状态的接口是活动的，可以用来接受网络流量。处于 down 状态的接口没有连接到网络。

你能配置什么类型的接口？现如今，时髦的计算机主机可以轻松拥有 1 个以上的网卡（或网络接口），有时还有几百个 IP 地址（或 IP 别名）。一个接口可以有許多别名，或者可以结合两个或更多接口以一个接口的形式出现。

IP 别名提供了在单个接口上拥有多个 IP 地址的能力。这种接口的所有 IP 地址都使用同一个 MAC 地址。一个结合式的或者说结队（teamed）的接口由两个或更多网络接口组成，形成单个接口。它可以用来提供更大的容错能力或增加网口的带宽。一个结合式的接口也可以有许多 IP 别名。这一点会在“Red Hat 的网络配置文件”和“Ubuntu 的网络配置文件”两节深入探讨。

每个网络接口大概至少都要分配 IP 地址。我们将从演示一个简单的工具开始介绍网口。这个工具叫作 `ifconfig`，可用来查看和改变接口的状态和配置。`ifconfig` 工具很像 Microsoft Windows 上的 `ipconfig` 命令。

要显示一个主机上的所有接口的状态，使用下面这个命令。

```
$ ifconfig -a
```

运行带 `-a` 选项的 `ifconfig` 命令会显示主机上所有接口和它们当前的状态和配置。

为了更容易探查一个接口的配置，还可以显示单个接口，如下所示。

```
$ ifconfig eth0
Eth0 Link encap:Ethernet HWaddr 00:0C:29:3F:69:60
    inet addr:192.168.0.1 Bcast:192.168.1.255 Mask:255.255.255.0
    inet6 addr: fe80::20c:29ff:fe3f:6960/64 Scope:Link
    UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
    RX packets:79 errors:0 dropped:0 overruns:0 frame:0
    TX packets:88 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:1000
    RX bytes:19995 (19.5 KiB) TX bytes:20765 (20.2 KiB)
    Base address:0x1078 Memory: ec840000-ec860000
```

它显示了网络接口 `eth0` 的状态。它分配有一个 IP 地址 `192.168.0.1`，我们很快就会解释它。每个以太网网卡有一个唯一的硬件识别号，用来标识自己以及与其他以太网设备通信。它被称作媒介访问控制（Media Access Control，MAC）地址，在输出的第一行可以看到它（本例中是 `HWaddr 00:0C:29:3F:69:60`）。

接下来是一个 IPv6 地址，这里是 `inet6 addr: fe80::20c:29ff:fe3f:6960/64`，它是一个 MAC 地址派生的本地链路地址。

■注：有关 IPv6 的更多信息，见 <http://tldp.org/HOWTO/Linux+IPv6-HOWTO>。

这可以用来与其他使用无状态自动配置（stateless autoconfiguration）的主机通信。无状态自动配置常用在像 PDA 和移动电话这类设备上，它不需要复杂的基础架构就可以与本地网络上的其他设备通信。

■注：有关无状态自动配置和 IPv6 的更多信息，见 <http://www.ipv6.com/articles/general/Stateless-Auto-Configuration.htm>。

接下来一行是接口的状态。此接口的当前状态是 `UP`。这意味着此接口是活动的，如果配置合适，就可以接受流量。

接下来的选项是 `MTU`，或最大传输单元。`MTU` 是网络包的最大字节数；1500 是通常的默认值。

所返回的其余信息是一些统计计数器。这些计数器，特别是那些记录为错误、丢失和碰撞的包可以预示可能的问题。本章稍后的“一般网络故障诊断”一节会深入讨论网络故障的诊断。

让我们再看一看 `eth0` 接口。此接口当前没有分配 IP 地址（很快就会演示这一点）。但是

首先比较一下前面的输出与 `eth0` 状态为 `down` 时所显示的输出。

```
$ ifconfig eth0
eth0 Link encap:Ethernet HWaddr 00:0C:29:3F:69:60
    inet6 addr: fe80::20c:29ff:fe3f:6960/64 Scope:Link
    BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
    RX packets:79 errors:0 dropped:0 overruns:0 frame:0
    TX packets:88 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:1000
    RX bytes:19995 (19.5 KiB) TX bytes:20765 (20.2 KiB)
    Base address:0x1078 Memory: ec840000-ec860000
```

注意到除了第三行缺了一个 `UP`，其他都相同。它表明接口状态为 `down`，因而不能通信。发出下面的命令可以使接口回到 `up` 状态。

```
$ sudo ifconfig eth0 up
```

■注：该命令还有一个简单的别名叫作 `ifup`，可以像这样使用它：`ifup eth0`。

为了使接口处于 `down` 状态，要使用下面的命令。

```
$ sudo ifconfig eth0 down
```

或者使用它的别名命令 `ifdown`。

```
$ sudo ifdown eth0
```

你可以给接口手动分配 IP 地址，或者在主机或者网络服务重启时自动指派。假设主机刚重启过，那么安装时配置的 IP 地址就会立刻赋予 `eth0` 接口。在给此接口分配一个 IP 地址时，`eth0` 的输出看起来如下所示。

```
eth0 Link encap:Ethernet HWaddr 00:0C:29:3F:69:60
    inet addr:192.168.0.1 Bcast:192.168.1.255 Mask:255.255.255.0
    inet6 addr: fe80::20c:29ff:fe3f:6960/64 Scope:Link
    UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
    RX packets:99 errors:0 dropped:0 overruns:0 frame:0
    TX packets:92 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:1000
    RX bytes:23534 (22.9 KiB) TX bytes:22208 (21.6 KiB)
    Base address:0x1078 Memory: ec840000-ec860000
```

现在可以看到 `eth0` 有 `192.168.0.1` 的 IP 地址，它的状态为 `UP`。

`ifconfig` 命令还可以用来添加和删除接口的 IP 地址。下面是给 `eth0` 接口添加 IP 地址 `192.168.0.1` 的命令。

```
$ sudo ifconfig eth0 add 192.168.0.1
```

这里给 `ifconfig` 命令传递了 3 个选项：要操作的接口、要采取的动作和要使用的 IP 地址。

`ifconfig` 命令还可以用来从接口中删除 IP 地址，如下所示。

```
$ sudo ifconfig eth0 del 192.168.0.1
```

你还可以阅读 `ifconfig` 命令的 `man` 页探索它的其他用途。

不过，`ifconfig` 命令太简单了。它也就能做所介绍的这些事了，但是作为一个操作接口的

快速工具，它还是有用的。下面几节将详述如何使用命令行和 GUI 工具配置接口和网络。我们将特别关注如何使用另外一个命令操作接口，这个命令叫 `ip`。

■注：使用 `ifconfig` 命令对接口做的改动不能持续到下次启动；在重新启动后，将丢失所有的改动。我们很快就会说明如何把改动持久地应用于主机。

6.1.2 在 GUI 下配置接口

正常情况下，至少已经在安装过程中给接口配置过一次 IP 地址了。正如第 2 章所讲的那样，这个地址是下面中的一种。

- 静态 IP 地址。
- DHCP 指派的 IP 地址。

静态 IP 地址是为给主机提供一个特定 IP 地址而手动配置的地址。这通常是服务器和其他不希望其 IP 地址变动的主机的设置方式。通过 DHCP 或者叫域名主机配置协议指派的地址是从地址池中分配的。这通常是台式机和客户机的配置方式，因为保持它们的 IP 地址不变并不那么重要。

许多种方式都可以编辑现有的网络接口或者添加/删除一个新接口。如果刚开始使用 Linux，那么到现在为止，使用 GUI 更容易些。我们将说明如何通过 GUI 以及使用工作脚本本身来做到这一点。

1. 使用 Red Hat 的 GUI 配置网络

Red Hat 主机有一个强大的用户界面来管理接口和网络配置。用它可以往网络接口添加 IP 地址以及管理主机的 `/etc/host` 文件、DNS 设置和路由。路由（route）是数据包用来在别的网络上寻找其他主机的途径。其中一个主要的路由你可能熟悉，就是默认网关，或者默认路由。

■注：默认路由是所有不能为其找到方向的主机的网络数据包的目的地。路由器在网络之间担当网关的角色，用来把独立的网络连接在一起。如果默认的路由器不能找到目的主机，它会把网络数据包发给默认路由，这样继续下去，直到到达网络数据包的最终目的地。

有两种方式可以启动 GUI，一种是使用“系统”→“管理”→“网络”菜单，如图 6-3 所示，另外一种方式是在命令行里使用 `system-config-network` 命令。

它显示了当前所拥有的可配置接口。如图 6-4 所示，有一个以太网设备叫作 `eth0`，它是活动的。这是目前正在提供网络服务的网络设备。

通过这个窗口你可以编辑当前可用的设备，或者只是让那个设备不活动。你还可以添加新设备或者删除现有的设备。

在“硬件”标签里可以查看以太网网卡设备驱动程序的相关信息。IPsec 标签显示 IPsec 信息，允许配置接口上的 IPsec 隧道。DNS 标签显示 DNS 的资料，允许编辑 DNS 设置。通过“主机”标签可以查看和编辑与网络相关联的主机文件。



图 6-3 在 Red Hat 上启动网络配置工具

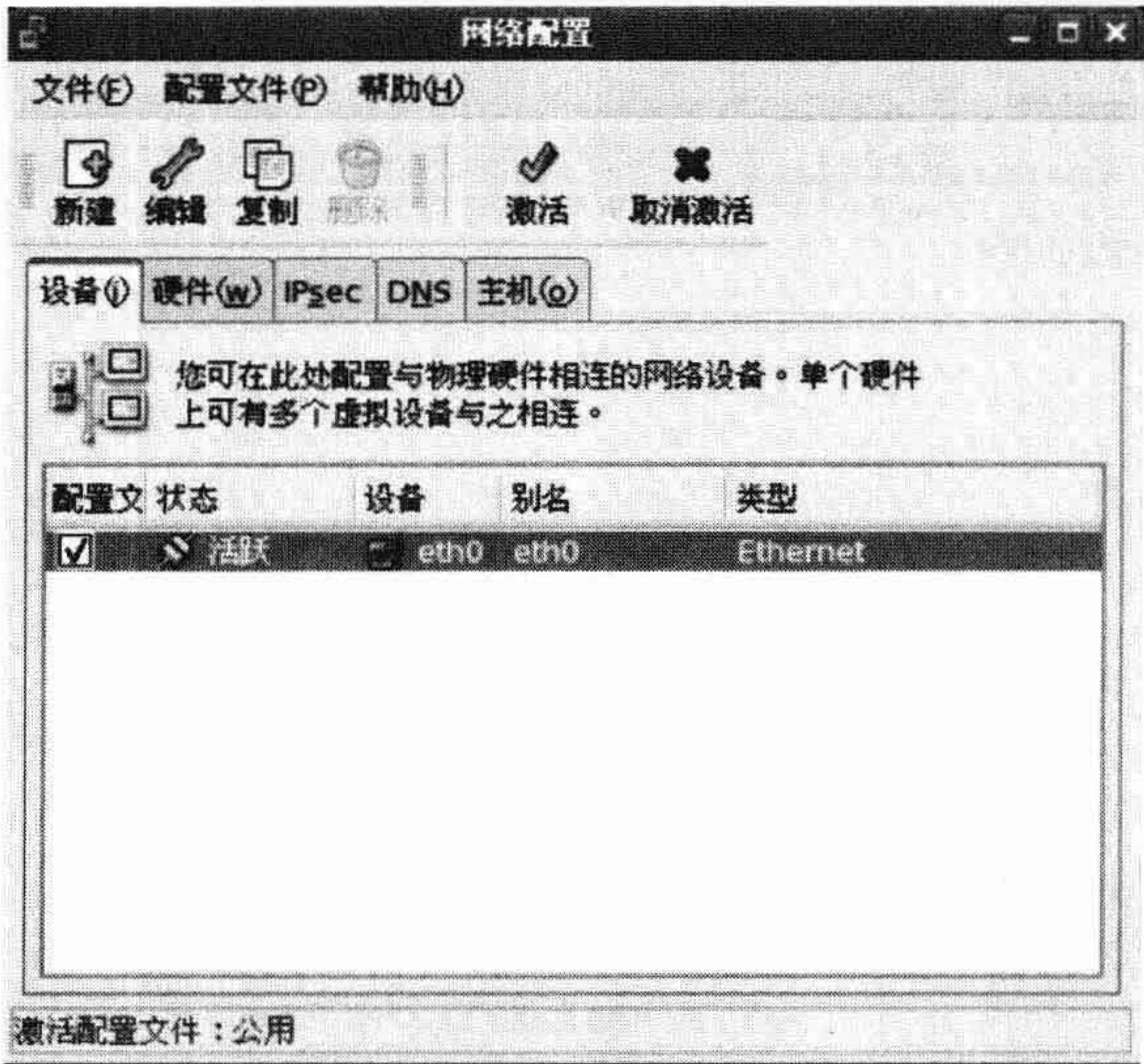


图 6-4 目前可用的网络设备

如果主机的网络接口多于 1 个，就可以添加一个新的以太网设备。单击“新建”按钮，显示完整的可创建设备类型列表，如图 6-5 所示。

可以看到，在这里可以创建几种不同类型的设备。我们关注于创建一个以太网设备。

注：本章后面的“配置范例网络”一节将演示创建一个连接到 Internet 的 xDSL 设备。

如图 6-6 所示，有两个以太网网卡可供选择。设备 eth0 已经配置好了，因此我们将选择第二个设备 eth1。

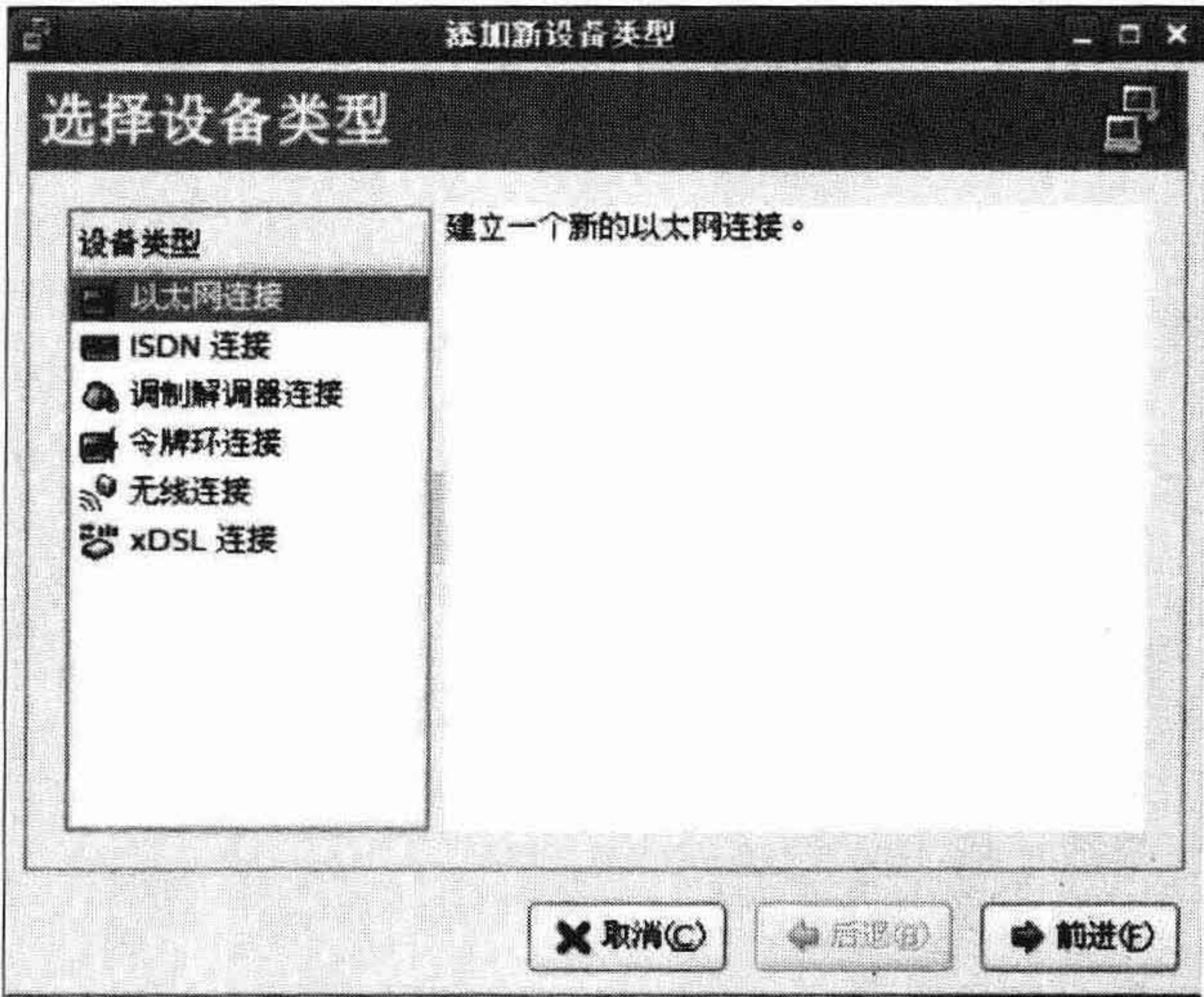


图 6-5 可创建的可能网络连接列表

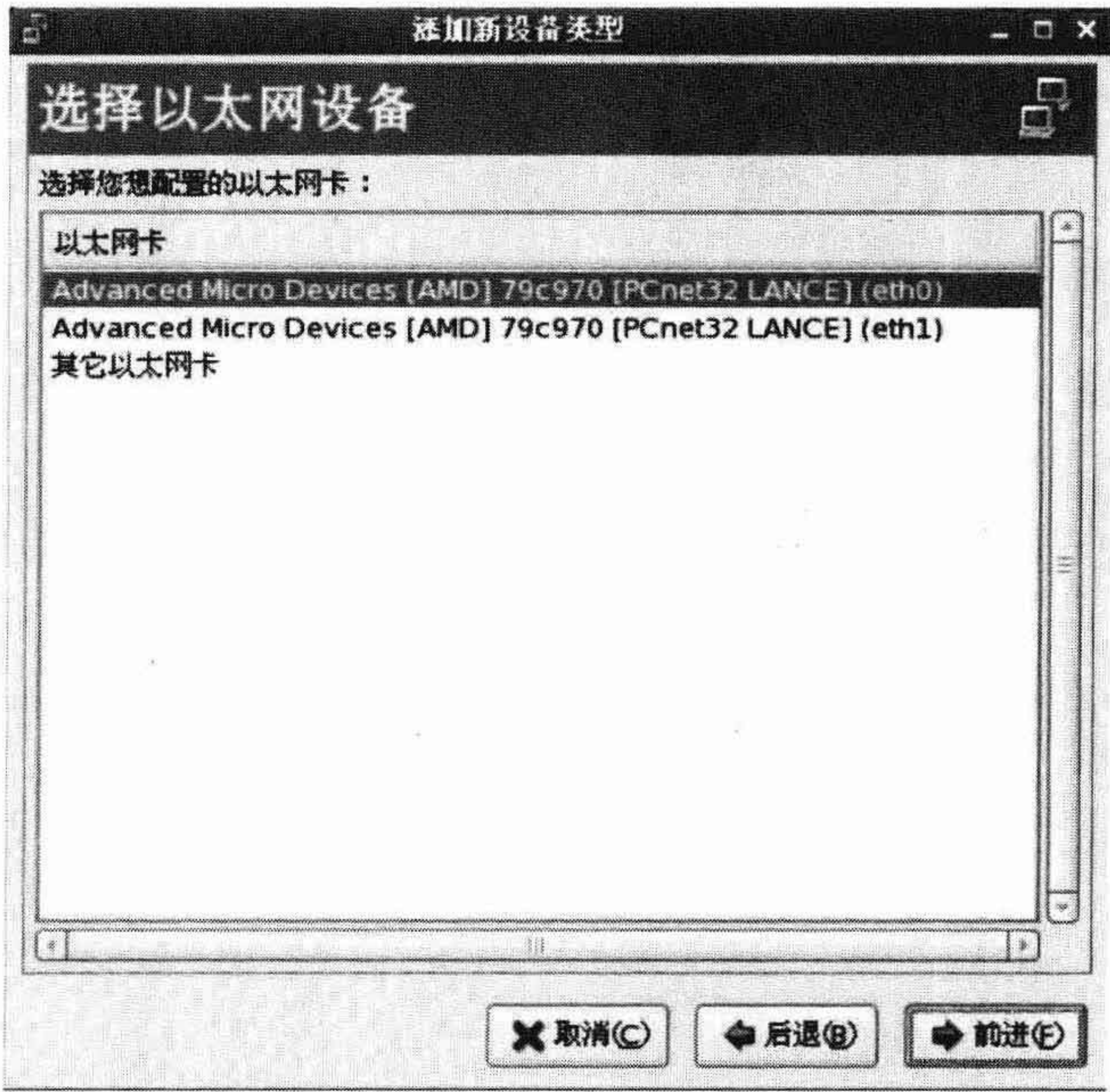


图 6-6 选择以太网设备

对于主机所拥有的每个网卡（或者虚拟网卡，如果正在使用一个如 VMware 这样的虚拟主机的话），都会在这里列出对应的设备。如果没有列出来，那么可以尝试使用“其他以太网卡”选项配置它，前提是知道它正常工作所需的驱动程序。内核必须连同正确的网卡驱动程序一起编译，它才能工作。它会呈现一个内核所知的以太网驱动程序列表。应该挑选与以太网设备最匹配的驱动程序。

接下来它会要求提供接口的 IP 地址资料，或者选择可使接口检索到这些资料的方式。从图 6-7 可以看到可用选项。

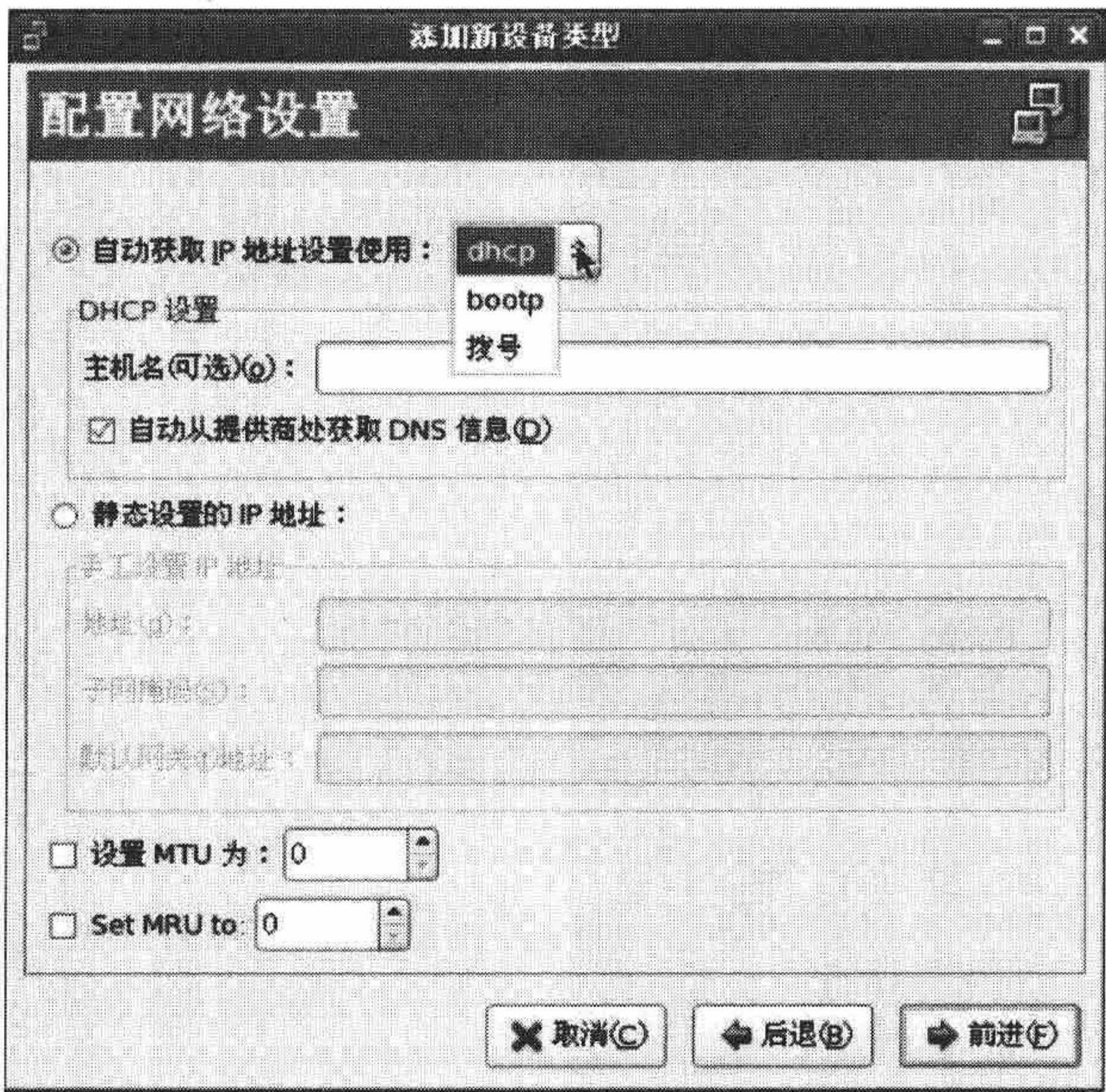


图 6-7 选择 IP 地址提供者

你可以从下面的选项中选择。

- dhcp
- bootp
- dialup

如前所述，DHCP 是一个接口用来请求从可用地址池指派 IP 地址的协议。它要求有一个可用的 DHCP 服务器来给它发放 IP 地址。许多 SOHO 路由器和 DSL 调制解调器都有内置的 DHCP 服务器，网络上的主机就由这些设备自动分配 IP 地址。

■注：第 9 章将讲述 DHCP 以及如何运行自己的 DHCP 服务器。

BOOTP 协议通常用于无磁盘工作站，使主机能够获得一个 IP 地址和一个操作系统镜像。然后它使用这个镜像启动设备。瘦客户端（例如无磁盘工作站和像超市收银机那样的服务终端机）没有硬盘驱动器来载入它们的操作系统。它们改为使用 BOOTP 协议检索操作系统镜像以及所有必需的网络信息。

最后一个选项是 dialup，用于通过公共交换电话网（Public Switched Telephone Network, PSTN）的电话线而进行的拨号上网调制解调器连接。这种连接如今已很少使用，因为大部分机构都有 DSL 或其他类型的宽带连接。

你还可以选择输入自己的地址信息，就是较早前讲过的静态 IP 地址，如图 6-8 所示。这里添加了一个 IP 地址 192.168.0.253。这个地址来自图 6-1 所示的范例网络。它从标记为可静态分配的 IP 地址列表中选择，而不能从 DHCP 所使用的地址池中选择。

■注：有关 DHCP 地址池的资料，请参阅第 9 章。

单击“前进”按钮之后，它呈现出已选择的选项概述。如果希望应用这些改动，单击“应

用”；如果不希望这么做，单击“取消”。

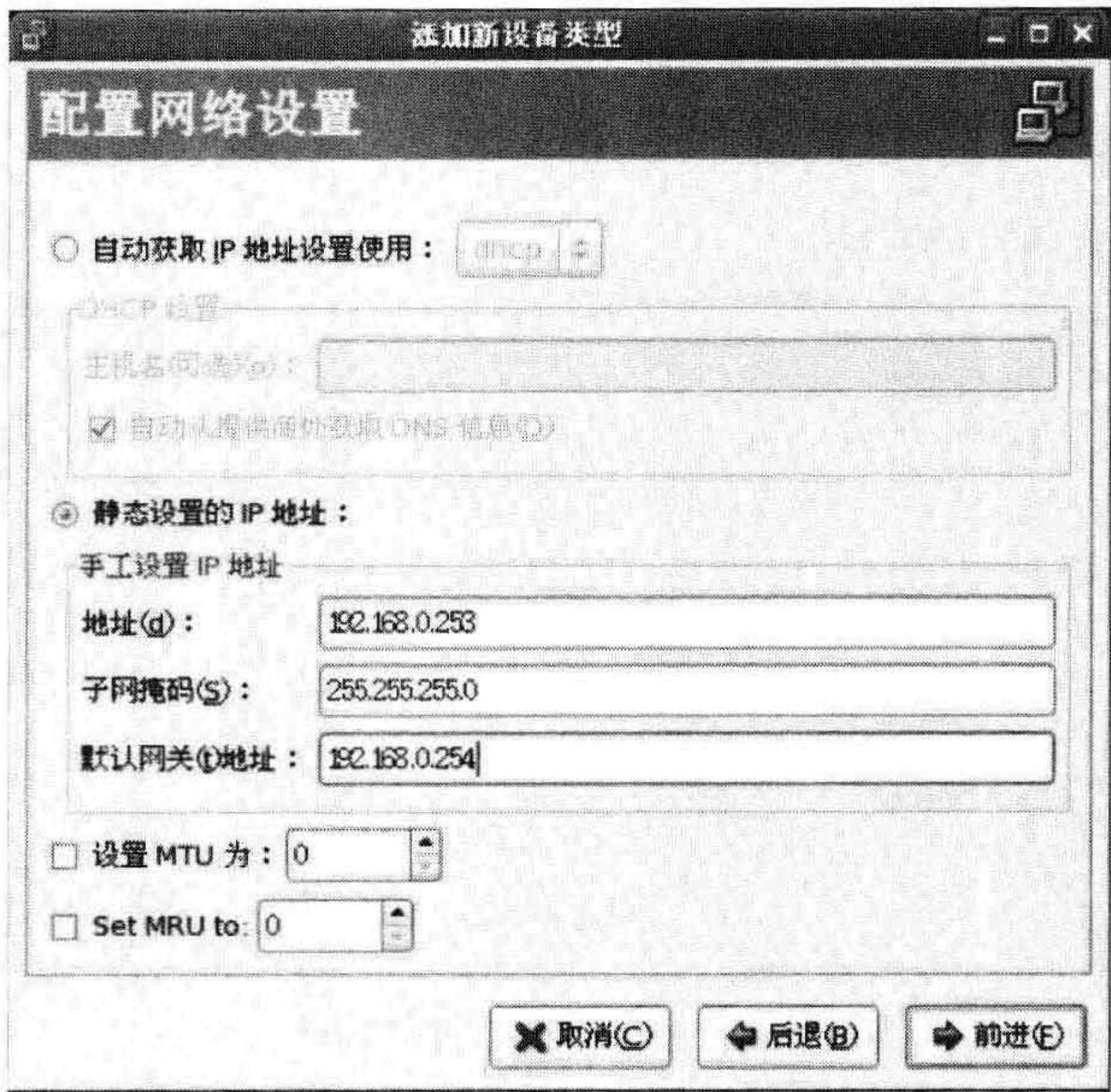


图 6-8 输入静态 IP 地址信息

如果单击了“应用”，马上就可以看到新以太网接口出现在列表中，如图 6-9 所示。

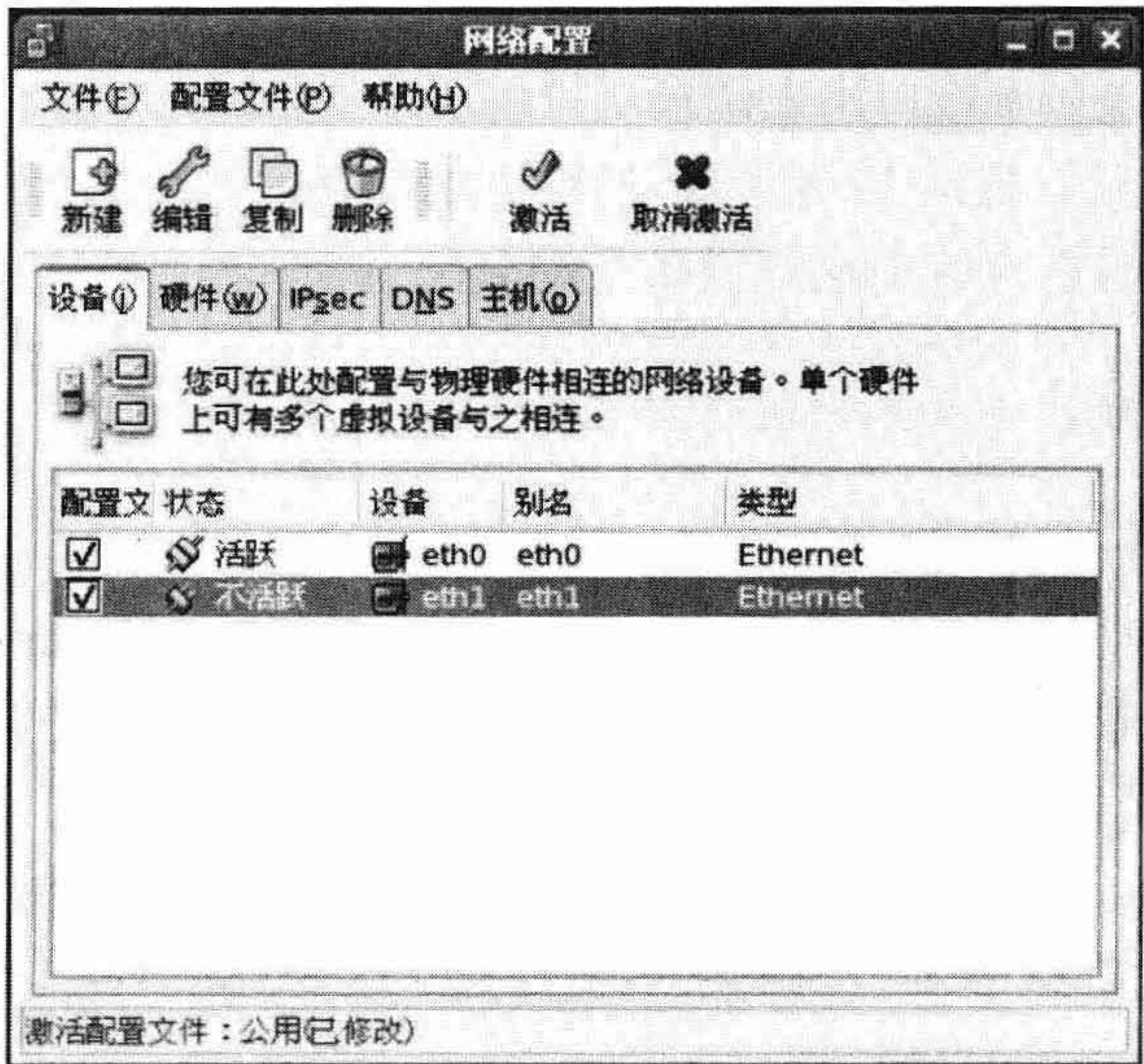


图 6-9 新配置的以太网接口

可以看到新设备是不活动的。在让它活动之前，必须先保存配置。为此，选择“文件”→“保存”，如图 6-10 所示。

它会显示一个报告画面，表明也许要重启网络服务或重启主机来激活新设备。Linux 不需重启就可处理新以太网设备的添加任务，并且 GUI 工具会在激活新设备时重启网络服务。

你还可以通过如下的命令重启网络服务。


```
$ sudo /sbin/service network restart
```

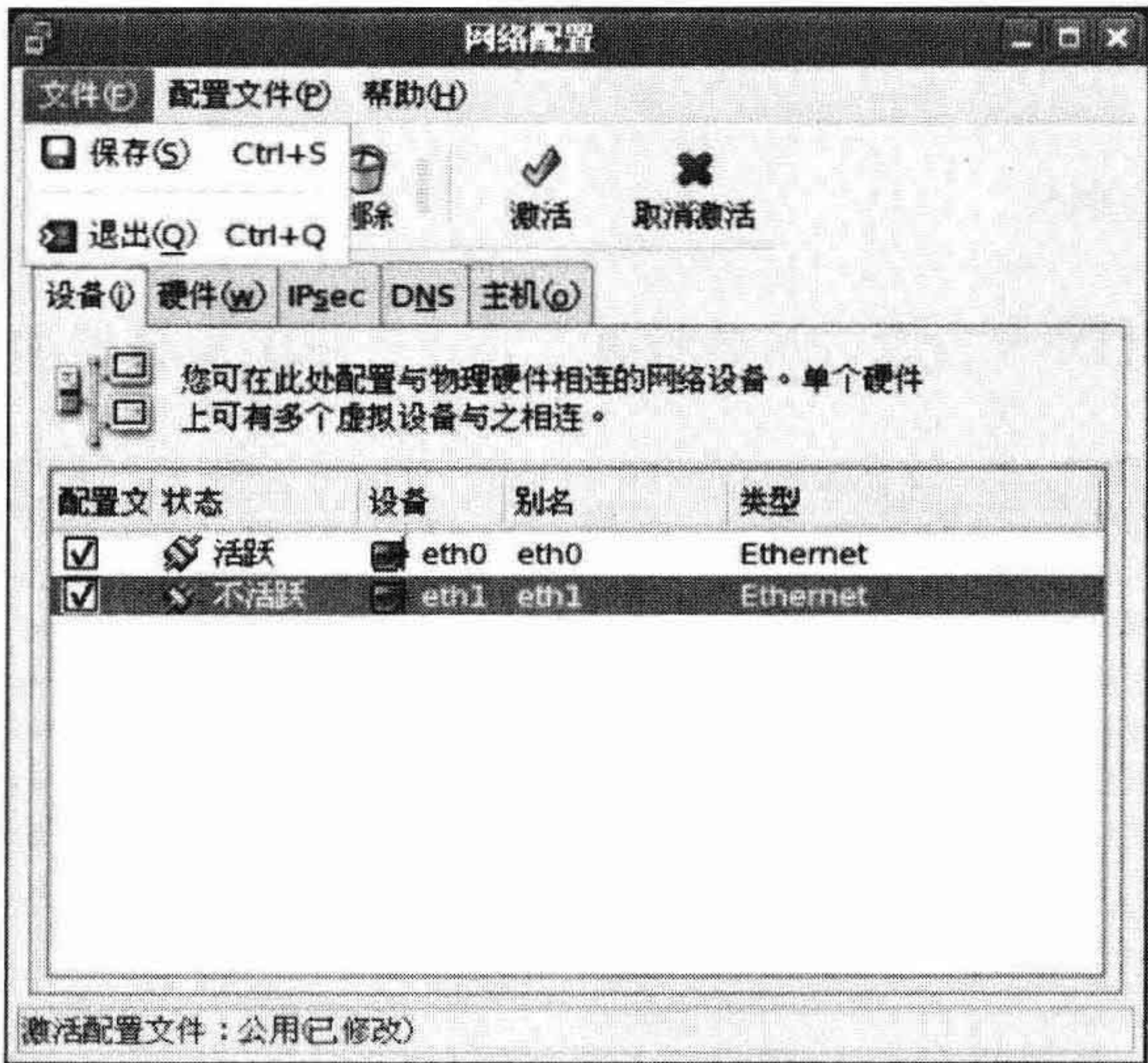


图 6-10 保存配置

要通过 GUI 做此事，就要突出显示新设备并单击“激活”按钮；呈现的画面显示激活的进度，然后就会显示两个设备都是活动的，如图 6-11 所示。

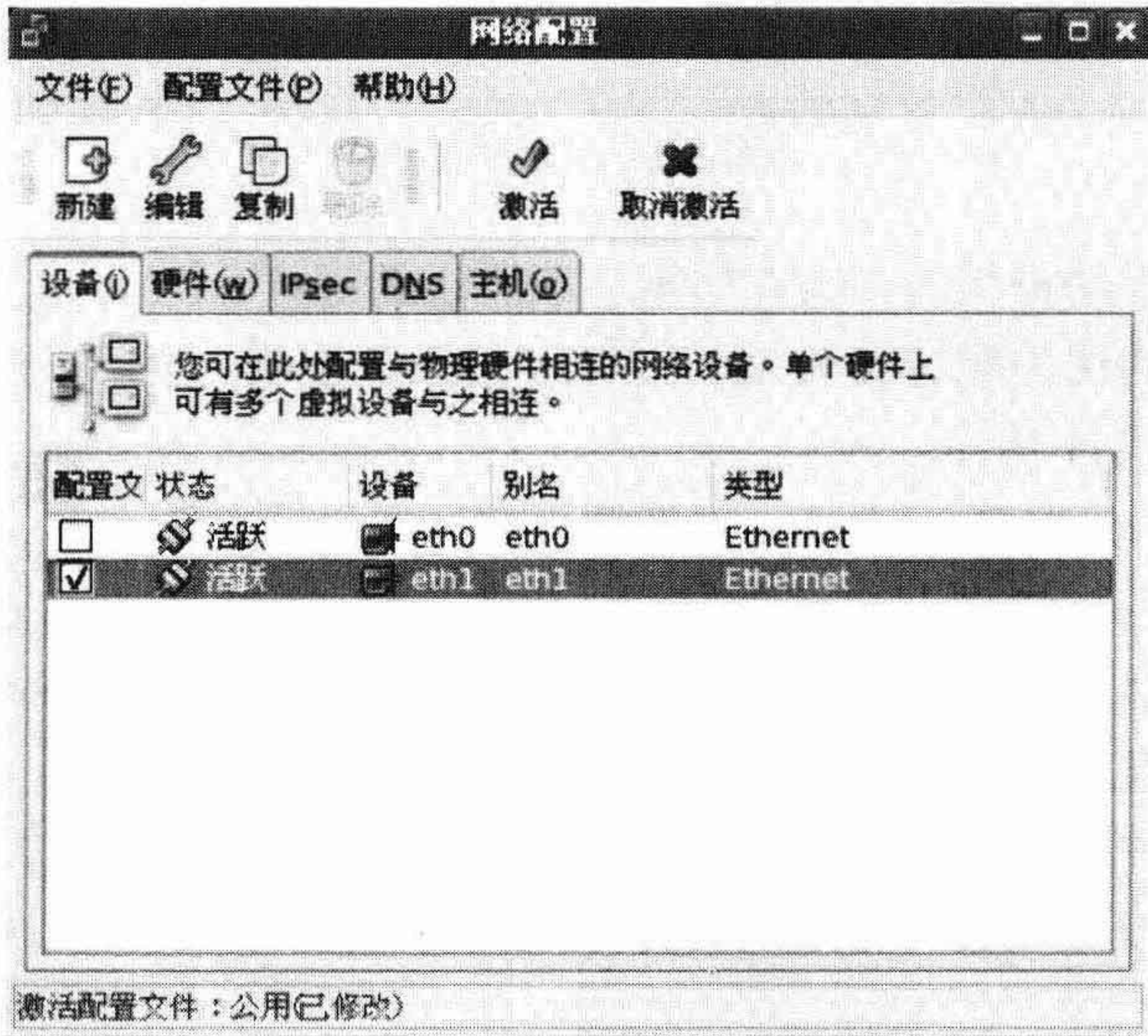


图 6-11 两个活动的以太网设备

现在有了两个活动的以太网设备。你将希望确保这些设备都在工作，这件任务很快就会介绍，不过首先让我们研究一下用这个 GUI 工具可做的其他事情。

从图 6-11 可以看到有两个活动的以太网设备 eth0 和 eth1。如果愿意的话，你可以编辑这些设备的配置设置。

首先让我们看看 DNS 设置。第 2 章在说明如何安装主机时简要介绍了 DNS。DNS 被

主机用来匹配主机名（如 au-mel-ubuntu-1.example.com）和与之关联的 IP 地址。通过向域名服务器发送一个 DNS 查询可以完成匹配任务。域名服务器信息存储在/etc/resolv.conf 文件里。

提示：第 9 章将详述 DNS。

从图 6-12 可以看到 DNS 设置。这些设置显示了主域名服务器和第二域名服务器。

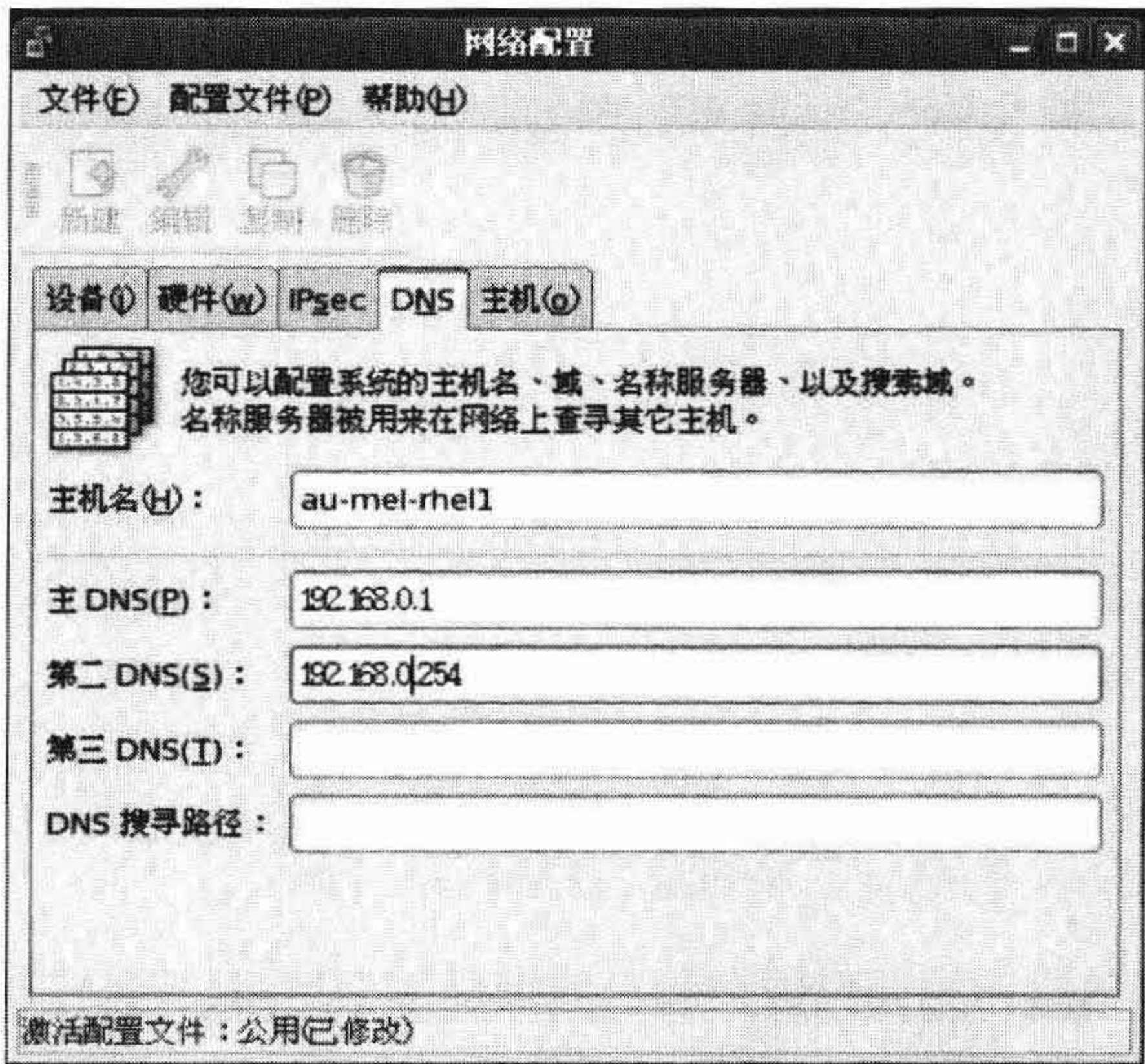


图 6-12 DNS 设置

这些设置或者是手动提供的，或者如第 9 章所讲的那样，由 DNS 服务器提供。这些服务器按照从最低到最高的优先顺序被查询。任何改动都要通过从菜单里选择“文件”→“保存”来保存，如前面的图 6-11 所示。

下一个可以修改的设置是/etc/hosts 文件的内容。该文件在希望提供特定主机的名字和 IP 地址信息时使用。例如，在 DNS 服务离线时不希望某两台主机停止通信，这时就适合使用它。但是它的缺点是如果地址信息改变了，你就必须在这台机器上维护这个信息。要在主机文件里增加一个主机索引，可移动到“主机”标签，单击“新建”按钮，输入相应的信息，如图 6-13 所示。

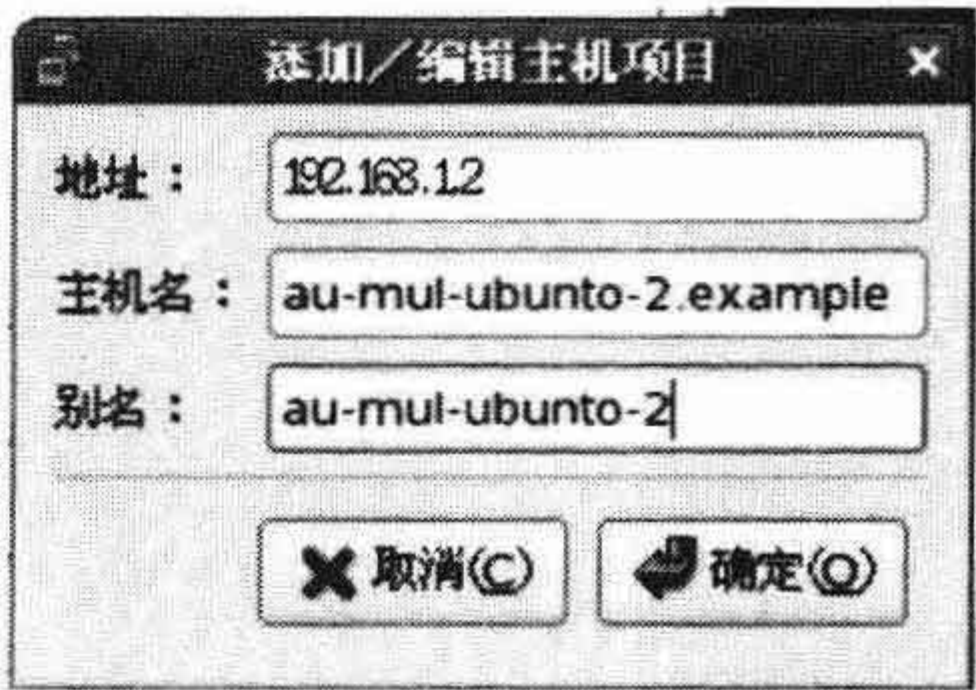


图 6-13 编辑 hosts 文件信息

注：编辑/etc/hosts 文件同样可以做到这一点。

这里添加了一个 Ubuntu 主机 au-mel-ubuntu-2 的资料，提供了 IP 地址、主机名全称 au-mel-ubuntu-2.example.com 和同样可用来指代它的别名 au-mel-ubuntu-2。那么现在如果失去了网络的 DNS 服务，主机仍然能够找到 au-mel-ubuntu-2 的正确 IP 地址——只要不改动那台 Ubuntu 主机的 IP 地址。

使用 GUI 也可以添加 IP 地址别名。IP 地址别名允许对单个接口设备分配或绑定 1 个以

上的 IP 地址。不过 IP 地址不支持 DHCP，因此必须为每个别名分配静态 IP 地址。

■注：事实上使用 system-config-network 工具创建 IP 别名比使用命令行更容易，不过添加非持久的 IP 别名还是通过命令行更容易，因为使用命令行工具对接口做的改动将在重启后失去作用。

为了用网络配置工具创建 IP 别名，打开“设备”标签，单击“新建”按钮创建新设备。选择以太网连接作为设备类型，如创建新设备 new1 时所做的那样。接下来，选择希望这个 IP 别名所赋予的以太网卡。本例中选择了 eth1 所在的以太网卡。

接下来，它会要求给别名设备分配 IP 地址。你可以基于不从 DHCP 地址池中选取的原则选择任一可用地址。你还需要添加默认网关和子网掩码。

第 2 章介绍了默认网关。而子网掩码用于把一个网络细分（叫做划分子网）为更小的部分。一个 255.255.255.0 的子网掩码拥有 254 个可用的寻址节点，而相比之下，255.255.255.248 的子网掩码只有 6 个可寻址节点。

■注：到 <http://en.wikipedia.org/wiki/Subnetwork> 可以了解更多有关子网的知识。

默认网关是网络上的一个主机，可以把流量路由到其他网络，如 Internet。我们的例子中选择了 192.168.0.23 并提供了如图 6-14 所示的资料。

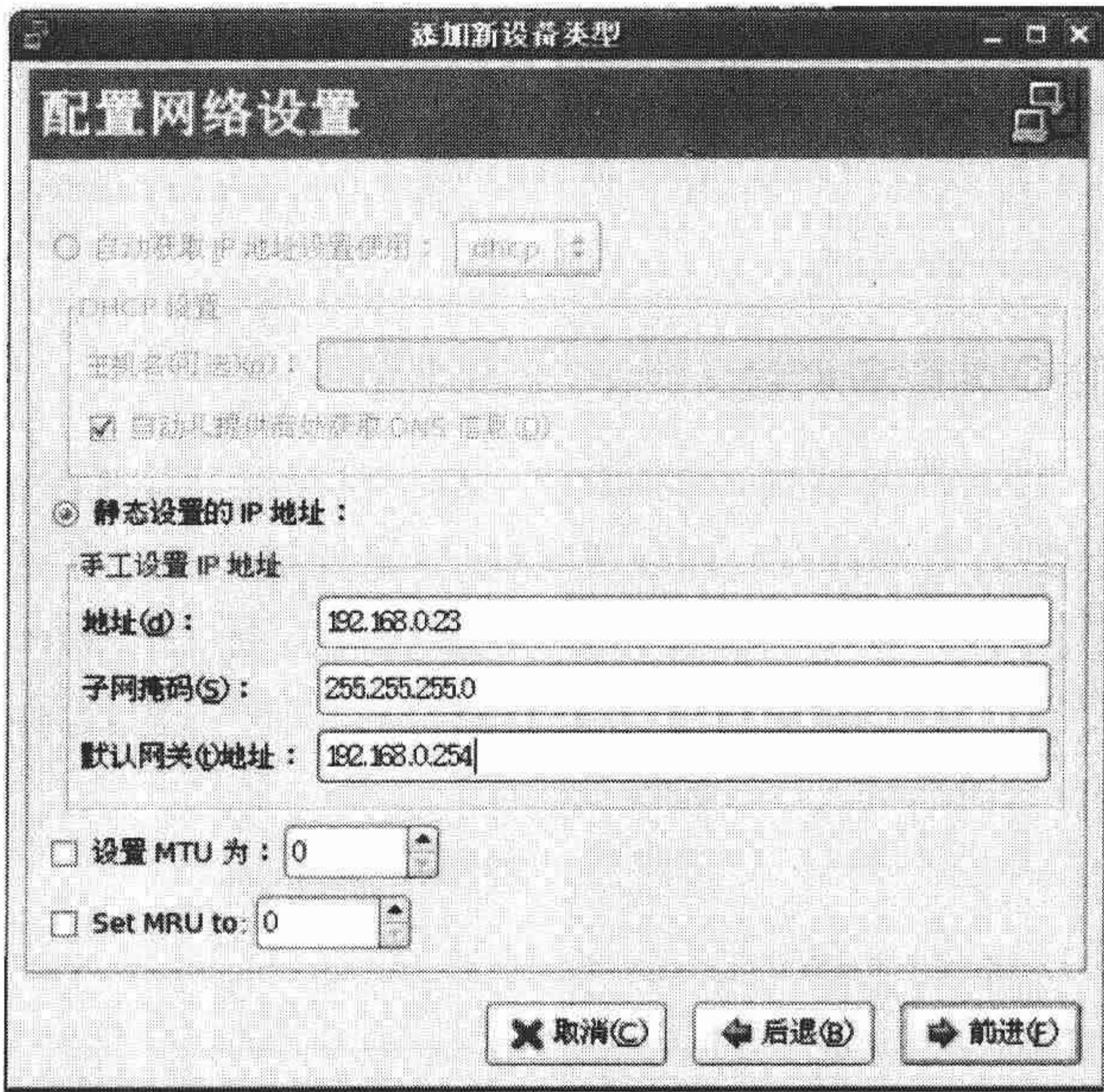


图 6-14 IP 别名配置

最后，让我们看看接口路由的添加。为了添加路由，我们进入网络配置工具里的“设备”标签并突出显示已创建设备。然后单击“编辑”按钮。可以看到一个“常规”标签，挨着它有一个“路由”标签。单击“路由”标签。如前所述，路由是主机用来寻找其他主机的途径。要为接口添加一个路由，单击“添加”按钮。从图 6-15 看看做了什么。



图 6-15 向接口添加路由

我们通过 192.168.0.2 主机为 eth0 添加了一个到 192.168.10.0/24 网络的路由。这样就在 /etc/sysconfig/network-scripts、/etc/sysconfig/networking/profiles/default 和 /etc/sysconfig/networking/devices 目录里创建了 3 个叫作 route-eth0 的配置文件。本章稍后“添加路由与转递网络包”一节将详细说明通过命令行查看和添加路由的方法。

Red Hat 主机里的所有网络设备配置文件都存储在 /etc/system/network-scripts 目录里，你也可以到那里查看或编辑这些文件。

2. 使用 Ubuntu 的 GUI 配置网络

Ubuntu 主机也有一个配置网络接口的 GUI 工具。这个工具 network-admin 是为了编辑现有的接口而不是创建新接口而设计的，而且使用它不能创建 PPP（点对点协议，Point-to-Point Protocol）连接。PPP 常用于在私人网络和 Internet 服务提供商（Internet service provider, ISP）之间建立一个直通的连接。你可以使用它编辑 /etc/hosts 文件，通过 /etc/resolv.conf 文件编辑 DNS，以及通过编辑 /etc/hostname 文件编辑主机名。

这个 Ubuntu GUI 工具也不是默认安装的，需要先安装它。为此，要使用 aptitude 命令。

```
$ sudo aptitude install gnome-network-admin
```

■注：第 7 章将详述软件包的安装。

通过使用与 Red Hat 相同的菜单路径 System→Administration→Network 开启它，或者在命令行里使用 network-admin 命令启动它。

```
$ sudo network-admin
```

改变网络设置需要 root 特权。注意图 6-16 中 Network Setting 窗口底端面板里的“Unlock”

按钮。单击该按钮后，它会要求提供一个密码。“Unlock”按钮运行 `sudo` 命令以便提供编辑网络配置的特权。

可以很自然地发现，它看起来像 Red Hat 里的 `system-config-network` 界面。你首先会看到一个不同类型的连接列表。例如，每个有线的连接都以连接类型和接口的形式列出。

你还会看到下面的标签：General，在这里可以编辑主机和域名；DNS，可编辑 DNS 配置；Hosts，在这里可往 `/etc/hosts` 文件添加主机。

通过双击“Wired connection”项（或者单击“Properties”按钮），你可以编辑此连接的属性。这样就会产生一个类似于图 6-17 的窗口。

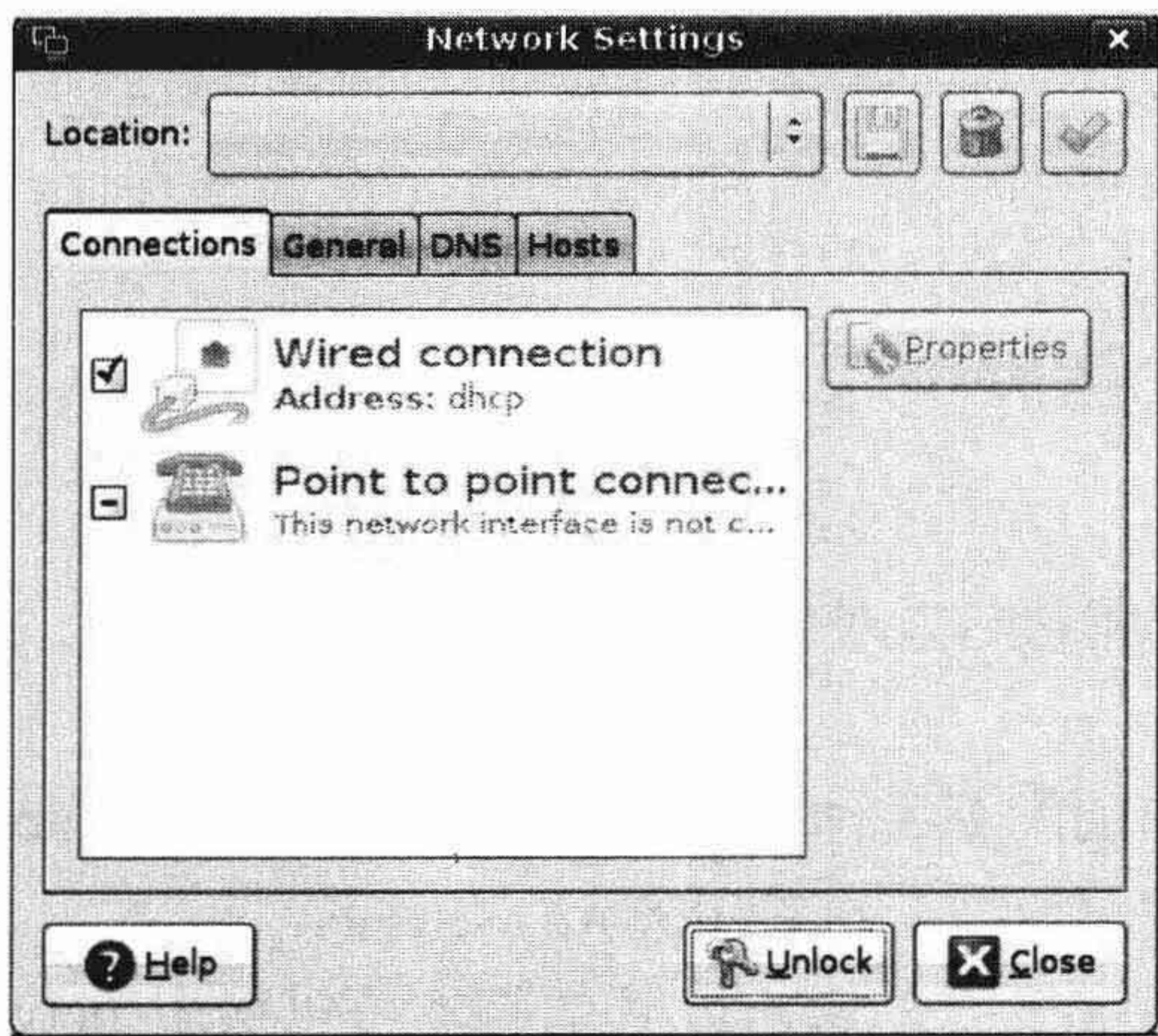


图 6-16 为 Ubuntu 网络管理工具解锁

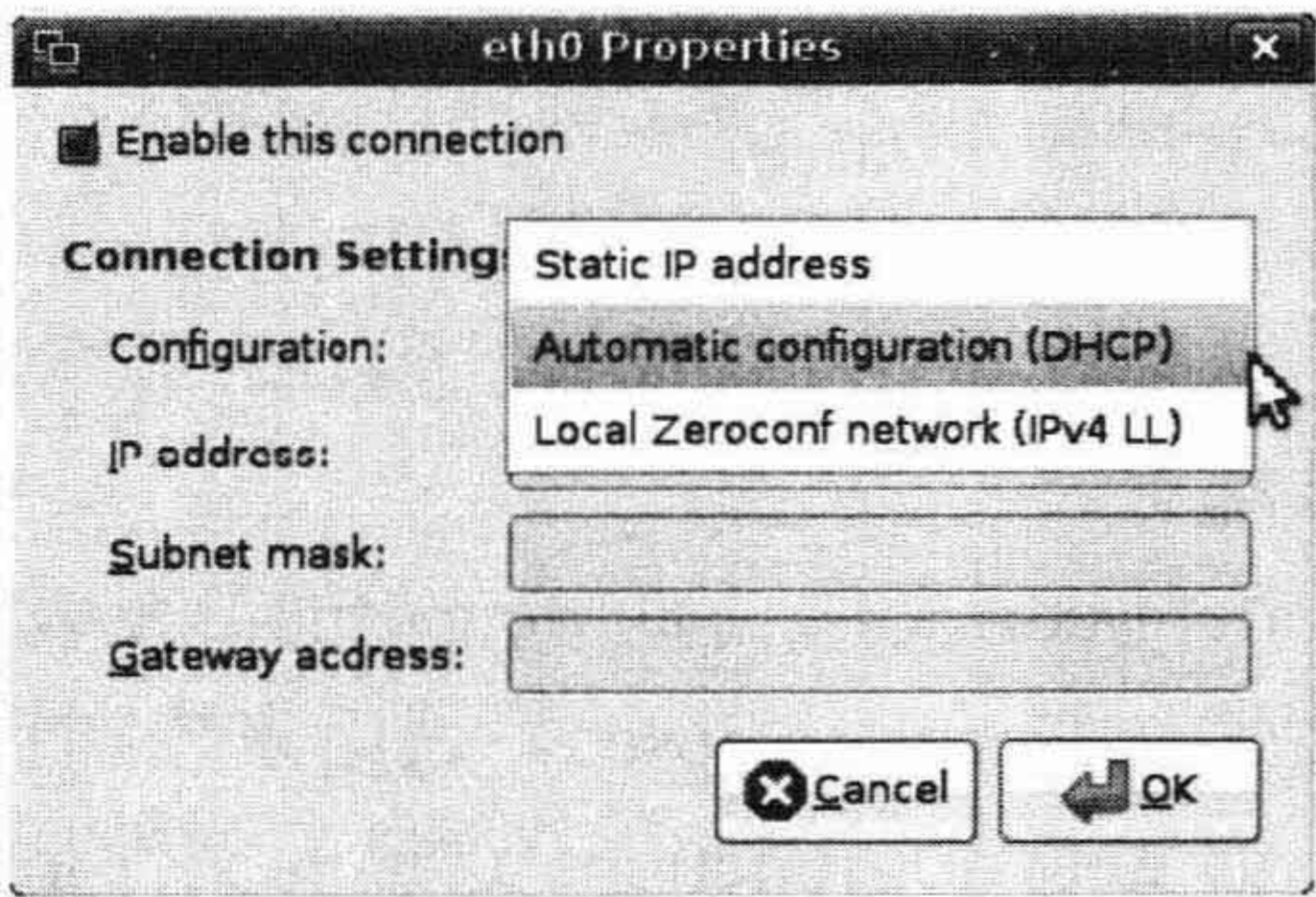


图 6-17 编辑 Wired connection 接口

如图 6-17 所示，你可以选择主机从 DHCP 服务器获得 IP 地址，或者配置为静态分配一个地址。

Ubuntu 网络设备的所有配置资料都存储在 `/etc/network/interfaces` 文件里。你还可以直接编辑该文件来添加更多的配置选项。这一点会在下一节详细探讨。

6.1.3 使用网络脚本配置网络

Red Hat 和 Ubuntu 两者都把它们网络配置文件存储在 `/etc` 目录里。Red Hat 把这些文件存储在 `/etc/sysconfig/network-scripts` 下与网络相关的目录集合里，而 Ubuntu 把这些文件全都存储在 `/etc/networks` 目录里。

1. Red Hat 的网络配置文件

Red Hat 上与网络相关的文件可以在 `/etc/sysconfig` 目录下找到。找寻这个信息的主要位置如下所示。

- `/etc/sysconfig/network` 文件
- `/etc/sysconfig/network-scripts` 目录
- `/etc/sysconfig/networking` 目录

/etc/sysconfig/network 文件包含一般的网络信息，如 HOSTNAME 和默认 GATEWAY。
/etc/sysconfig/network-scripts 目录包含网络接口的所有启动和关闭脚本。
/etc/sysconfig/networking 目录包含针对设备和针对配置档案（profile）的文件。这些文件是/etc/sysconfig/network-scripts 目录里的文件的通用副本，是 system-config-network 工具创建的。

看一看/etc/sysconfig/network-scripts 目录里的内容，可以看到如列表 6-1 所示的文件。

列表 6-1 /etc/sysconfig/network-scripts 里的文件

```
$ sudo ls /etc/sysconfig/network-scripts/
ifcfg-eth0 ifdown-ippv ifup-bnep ifup-sit
ifcfg-eth1 ifdown-ipsec ifup-eth ifup-sl
ifdown-ipv6 ifup-ippv ifup-tunnel ifdown-isdn
ifup-ipsec ifup-wireless ifdown-post ifup-ipv6
init.ipv6-global ifdown-ppp ifup-ipx ifdown-routes
ifup-isdn ifdown-sit ifup-plip net.hotplug
ifcfg-lo ifdown-sl ifup-plusb network-functions
ifdown ifdown-tunnel ifup-post network-functions-ipv6
ifdown-bnep ifup ifup-ppp ifdown-eth
ifup-aliases ifup-routes
```

从列表 6-1 可以看到，所有的脚本都用来配置接口并使它们处于 up 或 down 状态。各种文件都有。

ifcfg-eth?文件（如 ifcfg-eth0）是以太网接口的配置文件。符合命名规则 ifaction-interface 的文件（如 ifdown-ipsec）是用来控制状态（就是使接口 up 或 down）的脚本。

如 network-functions 这样的文件是包含函数和变量的脚本。其他脚本可以援引 network-functions 的函数和变量，并在它们的脚本中使用。

补充知识：援引其他脚本的函数和变量

当在 Bash 里写程序时，就像许多语言那样，你可以把想要与其他脚本共用的函数、变量和实用工具包含在一个脚本中，以节省重写同样代码的工夫。例如，假定有一个处理程序中错误的函数。对于所创建的每个脚本，你可以重写相同的错误处理函数，或者改为把它写入一个脚本，然后把它作为其他脚本的源，或者把它导入到其他脚本。

如果把错误处理脚本命名为 errors.sh，那么就可以把下面一行添加到新脚本里。

```
#!/bin/bash
. ./errors.sh
```

第一个句点 (.) 跟着一个空白空格，这是写 source 命令的快捷键。source 命令会把 errors.sh 的函数和变量导入到当前脚本。这样就节省了时间和能量，避免了复制脚本里的所有函数。

让我们看一看 eth0 的配置文件，这个文件被称作 ifcfg-eth0。从列表 6-2 可以看到该文件的内容。

列表 6-2 ifcfg-eth0 文件

```
$ sudo less /etc/sysconfig/network-scripts/ifcfg-eth0
# Intel Corporation 82545EM Gigabit Ethernet Controller (Copper)
DEVICE=eth0
BOOTPROTO=dhcp
HWADDR=00:0c:29:3f:69:56
```



```
ONBOOT=yes
TYPE=Ethernet
USERCTL=no
IPV6INIT=no
PEERDNS=yes
```

从中可以看到许多格式为 option=argument 的配置选项。

从列表 6-2 可以看到，为了配置一个 Red Hat 接口，要像这样指定设备名：DEVICE=eth0。启动协议 BOOTPROTO=dhcp 被设置为从 DHCP 获得地址。设备的 MAC 地址是 HWADDR=00:0C:29:3f:69:56，这是分配给这块网卡的唯一标识符。我们还通过指定 ONBOOT=yes 声明接口在启动时是否初始化。接口类型被声明为 TYPE=Ethernet。然后使用 USERCTL=no 声明用户是否可以开启或停止此接口。因为不想用 IPv6 的地址初始化接口，所以我们明确声明 IPV6INIT=no。我们还设置 PEERDNS=yes。在声明 PEERDNS=yes 时，我们希望使用 DHCP 服务器所提供的域名服务器修改/etc/resolv.conf 文件。如果把它设置为 no，那么当此接口为 up 状态时，/etc/resolv.conf 保持不变。

表 6-1 列出了在 Red Hat 接口文件中可以使用的选项。

表 6-1 Red Hat 网络配置文件选项	
选项	说明
DEVICE	正在创建的设备名称。它会出现现在接口列表中
BOOTPROTO	设备启动时使用的协议。它的选项是 static、dhcp 和 none
ONBOOT	主机启动时设备是否开启
NETWORK	此设备的网络地址
NETMASK	此设备的子网掩码
IPADDR	此设备的 IP 地址
USERCTL	某个用户是否可开启或停止设备。选项是 yes 或 no
MASTER	此设备为 SLAVE 时对应的那个设备
SLAVE	此设备是否被 MASTER 指令中指定的设备所控制
ETHTOOL_OPTS	ethtool 的次序依赖选项。在可全双工的设备上设置全双工或设置速度和协商参数时 有用
DNS	DNS 主机的 IP 地址（多个地址用逗号分开）。如果 PEERDNS 设置为 yes，那么该地 址会添加到/etc/resolv.conf 中
PEERDNS	决定 DNS 选项中定义的 DNS 主机是否添加到/etc/resolv.conf。如果设置为 yes，就添 加，否则不添加

现在要做的是用这个信息说明如何设置一个结合式以太网设备。一个结合式以太网设备也可称为干线（trunk）设备。结合的方式可以把两个或更多以太网端口当一个接口用，从而获得扩展的带宽和一些冗余度。这样就可以使一个虚拟的接口从 1GiB 的链接变成一个 2GiB 的链接。

那么如何实现这个技巧呢？首先编辑用 system-config-network GUI 创建的 eth0 和 eth1 配置文件，如列表 6-3 所示。

列表 6-3 eth0 从设备配置

```
$ less /etc/sysconfig/network-scripts/ifcfg-eth0
# Intel Corporation 82545EM Gigabit Ethernet Controller (Copper)
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=none
USERCTL=no
SLAVE=yes
MASTER=bond0
```

这是一个很简单的配置，指定了想要控制的设备 eth0，并通过 ONBOOT=yes 指定了是否在主机启动时初始化它。这个结合式设备的 IP 地址是 bond0 设备的地址，不是 eth0 或 eth1 的地址；因此，这里没有指定启动协议，而是使用了选项 BOOTPROTO=none。

我们还不想让用户控制此接口的状态是 up 还是 down，因而设置了 USERCTL=no。接下来是两个把此设备添加到一个结合式接口的选项。第一个 SLAVE=yes 声明此设备是从设备。然后通过 MASTER=bond0 声明它属于哪个主设备。所提到的 bond0 是与即将创建的设备同名的设备。

对于接口 eth1（它在/etc/sysconfig/network-scripts/ifcfg-eth1 文件中配置），将以 eth0 为样板，从 eth0 粘贴资料。然后需把 DEVICE=eth0 改为 DEVICE=eth1。其他的内容保持一致。

接下来将创建 bond0 设备文件。在 Red Hat 里，配置资料会保管在一个叫作/etc/sysconfig/network-scripts/ifcfg-bond0 的文件里。列表 6-4 中给出了创建一个结合式以太网设备所需的东西。

列表 6-4 结合式以太网设备的配置

```
[jsmith@au-mel-rhel-1 ~]$ vi /etc/sysconfig/network-scripts/ifcfg-bond0
DEVICE=bond0
BOOTPROTO=none
ONBOOT=yes
NETWORK=192.168.0.0
NETMASK=255.255.255.0
IPADDR=192.168.0.1
USERCTL=no
TYPE=Ethernet
```

可以看到，它很像标准的以太网设备文件。你需要指定设备名（本例中为 bond0），给出如 IP 地址、网络和子网掩码这样的网络信息。此外我们希望此设备在启动时初始化，同时不希望其他用户可以控制它的 up 和 down 状态。

要把这些文件保存在/etc/sysconfig/network-scripts 目录里。你接下来要做的事是通知内核已为结合式设备准备好了新设备。

在本例中，通过在/etc/modprobe.conf 文件添加下面的命令行做到了这一点。

```
alias eth0 e1000
alias eth1 e1000
alias bond0 bonding
```

这告诉内核有了一个结合式设备，并希望称它 bond0。

为了增强性能，我们可以往结合式设备上添加参数。把这些参数添加到/etc/modules 文件。


```
bond0 mode=1 miimon=100
```

mode 1 把接口结合类型设置为活动备份。当活动的接口有问题时，另一个接口将会接管。**miimon** 选项指定多长时间检查一次接口的活动状态。在高效的配置里，当 **miimon** 注意到一个接口状态为 **down** 时，它会激活剩余的接口。

根据网络和给定的网络设备所能实现的组合类型的情况，你可以在这里添加其他选项，为结合式设备提供容错性、冗余度和循环调度（round-robin）功能。更多信息请到这里查阅 Red Hat 的手册：<http://www.redhat.com/docs/manuals/enterprise/RHEL-3-Manual/ref-guide/s1-modules-ethernet.html>。

你现在可以使用 **insmod** 命令插入这个模块并创建别名，如下所示。

```
sudo /sbin/insmod bond0
```

另外一个选择是重启主机。如果发出 **ifconfig** 或 **ip addr show** 命令，你就可以看到这两个设备成为了一个接口，本例中叫作 **bond0**。

通过发出 **/sbin/ip addr show** 命令可以查看新的结合式设备，它会产生下面的输出。

```
2: eth0: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> ↗
mtu 1500 qdisc pfifo_fast master bond0 qlen 1000
  link/ether 00:0c:29:3f:69:56 brd ff:ff:ff:ff:ff:ff
  inet6 fe80::20c:29ff:fe3f:6956/64 scope link
    valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> ↗
mtu 1500 qdisc pfifo_fast master bond0 qlen 1000
  link/ether 00:0c:29:3f:69:56 brd ff:ff:ff:ff:ff:ff
  inet6 fe80::20c:29ff:fe3f:6956/64 scope link
    valid_lft forever preferred_lft forever
5: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> ↗
mtu 1500 qdisc noqueue
  link/ether 00:0c:29:3f:69:56 brd ff:ff:ff:ff:ff:ff
  inet 192.168.0.1/24 brd 192.168.0.255 scope global bond0
  inet6 fe80::20c:29ff:fe3f:6956/64 scope link tentative
    valid_lft forever preferred_lft forever
```

如果看一看 **eth0** 和 **eth1** 的接口描述，可知两者都被设置为 **SLAVE**，而 **bond0** 设置为 **MASTER**。**eth0** 或 **eth1** 都没有相关联的 IP 地址。只有 **bond0** 接口才有相关联的 IP 地址。

如果希望为 Red Hat 主机上的接口创建设备别名，你可以做下面的事情。大家知道，所创建设备的所有配置文件必须使用一个名字类似 **ifcfg-eth1:1** 的设备文件存储于 **/etc/sysconfig/network-scripts** 目录。每个设备文件以它的设备名结束。**ifcfg-eth1:n** 表示这些文件属于 **eth1** 设备。**eth1** 设备上每个随后的别名设备会增加到 **n+1**。

别名文件看起来如下所示。

```
DEVICE=eth1:1
BOOTPROTO=none
TYPE=Ethernet
USERCTL=no
IPV6INIT=no
PEERDNS=yes
NETMASK=255.255.255.0
```



```
IPADDR=192.168.0.22
GATEWAY=192.168.0.254
ONPARENT=no
```

你可以看到这个文件除了 `DEVICE` 变量被定义为 `eth1:1` 之外，看起来很像一个常规设备文件。这里还有一个叫作 `ONPARENT` 的变量，它被设置为 `no`。`ONPARENT` 变量是别名设备特有的，它和标准设备文件中的 `ONBOOT` 设置相关。它决定别名设备是否随父设备（本例为 `eth1`）启动而启动。把它设置为 `no` 的意思就是它不会在父设备启动时自动开启。

另一种为以太网设备分配多个 IP 地址的方式是创建所谓的 `range` 配置文件。这是一个克隆文件，允许克隆一个以太网设备。它和别名设备有一点不同。克隆文件可以用来为接口指定附加的选项，它不能用于创建新设备，只可用来修改已存在的设备。为了创建一个 IP 地址范围（这和创建多个 IP 别名文件效果相同），我们要创建一个叫作 `/etc/sysconfig/network-scripts/ifcfg-eth0-range0` 的文件。在这个文件里添加了下面的选项，以便给 `eth0` 接口增加一个 IP 地址范围。

```
IPADDR_START=192.168.100.10
IPADDR_END=192.168.100.210
CLONENUM_START=10
NETMASK=255.255.255.0
```

可以看到，该文件中没有指定 `DEVICE`。这是因为它只是扩展了已存在的 `eth0` 设备文件。这个克隆文件在 `eth0` 设备上创建了一个从 `192.168.100.10` 到 `192.168.100.210` 的 IP 地址范围。以太网克隆设备从 `eth0:10` 开始，到 `eth0:210` 结束。（不一定要从 `eth10` 开始。如果愿意的话，可以从 `eth0:0` 或 `eth0:20` 开始。）

另一种添加别名的方式是用脚本在主机启动时分配 IP 地址。我们可以往 `/etc/rc.d/rc.local` 脚本（这是启动次序中最后一个运行的脚本，第 5 章提到过）中添加如下所示的内容。

```
/sbin/ip addr add 192.168.0.24/24 brd 192.168.0.255 dev eth1
/sbin/ip addr add 192.168.0.25/24 brd 192.168.0.255 dev eth1
/sbin/ip addr add 192.168.0.26/24 brd 192.168.0.255 dev eth1
```

我们很快就会讲解 `ip` 命令，不过该命令要做的事情是在主机启动时为 `eth1` 设备分配 3 个 IP 地址。

■注：eth0 这个名字是怎么回事？它实际上是内核用来控制设备的模块名别名（驱动程序是 Microsoft Windows 用户熟悉的术语）。你可以在 Red Hat 里的 `/etc/modprobe.conf` 或 Ubuntu 里的 `/etc/modprobe.d/aliases` 中找到以太网模块的别名。它们像这样表示：`alias eth0 e1000`。关于 Linux 内核模块的详细说明，请阅读 <http://tldp.org/HOWTO/Module-HOWTO/>。

2. Ubuntu 的网络配置文件

Ubuntu 里的 `/etc/network` 中有一个类似的网络文件目录。它包含 Ubuntu 用于设置网络的网络文件和脚本。正如较早所讲述的那样，Ubuntu 把接口信息存储在文件 `/etc/network/interfaces` 中。这个文件包含所有已配置过的接口信息。

使用 DHCP 的接口可以如此简单。


```
# The primary network interface
auto eth0
iface eth0 inet dhcp
```

首先用 `auto eth0` 声明 `eth0` 自动启动。接下来声明接口 `eth0` 使用从 DHCP 服务器获得的 IPv4 地址，即 `iface eth0 inet dhcp`。如果要配置以太网卡 `eth1` 或 `eth2`，同样要使用 `/etc/network/interfaces` 文件。

接口文件中可以使用的参数如表 6-2 所示。

表 6-2 /etc/network/interfaces 里的 Ubuntu 参数	
参数	说明
auto	使接口在启动时为 up 状态
inet	指定 IPv4 选址方式
inet6	指定 IPv6 选址方式
ppp	指定此设备为 PPP 连接
address	指定 IP 地址
netmask	指定子网掩码
gateway	为接口指定默认网关
dns-nameserver	为接口指定域名服务器
post-up	指定接口 up 后要执行的动作
pre-down	指定接口 down 之前要执行的动作

从表 6-2 可以看到在 Ubuntu 上设置网络接口时可以使用的大部分参数。我们要使用其中的一些参数在 `/etc/network/interfaces` 文件里为 `eth0` 设置一个静态网络接口，如下所示。

```
auto eth0
iface eth0 inet static
address 192.168.0.10
netmask 255.255.255.0
gateway 192.168.0.254
dns-nameservers 192.168.0.1
```

这里设置了 `eth0` 接口。它被设置为在主机启动时自动 up，即 `auto eth0`，并且告诉操作系统它分配了一个静态 IP 地址，即 `iface eth0 inet static`。并且为 `eth0` 接口提供了一个 `192.168.0.10` 的地址和一个 `192.168.0.254` 的默认网关（默认路由）。它还指定了 DNS 服务器 `192.168.0.1`（这是内部网络的第一域名服务器）。

现在用下面的例子说明如何使用它。通过使用接口文件，我们将在 Ubuntu 主机上创建一个结合式以太网设备。

我们要做的第一件事是安装另外一个软件包。

```
sudo aptitude install ifenslave
```

这个实用工具使得接口之间能够实现某种简单的轮询负载均衡，并且可以完成从设备的添加和分离。

接下来要把下面的内容追加到 `/etc/modules` 文件。


```
bonding mode=0 miimon=200
```

它告诉内核以模式 0 加载结合式的驱动器并每 200ms 监视一次接口。模式 0 是一种轮询负载均衡的配置，在这种配置下数据包依次通过每个接口传输。

现在需要配置这些接口。为此，编辑 Ubuntu 的 `/etc/network/interfaces` 文件以添加下面的内容。

```
# The primary network interface
auto bond0
iface bond0 inet static
address 192.168.0.10
netmask 255.255.255.0
gateway 192.168.0.254
dns-nameservers 192.168.0.1
post-up ifenslave bond0 eth0 eth1
pre-down ifenslave -d bond0 eth0 eth1
```

第一行 `auto bond0` 声明 `bond0` 设备要在启动时自动加载。接下来的一行 `iface bond0 inet static` 声明接口 `bond0` 是静态分配 IPv4 地址的接口，意思就是不使用 DHCP 或其他协议给它分配地址。然后使用关键词 `address`、`netmask`、`gateway` 和 `dns-nameservers` 分别分配 IP 地址、子网掩码、网关和 DNS 服务器。下面一行允许指定一个接口启动之后的命令。`ifenslave` 命令 `post-up ifenslave bond0 eth0 eth1` 在 `bond0` 状态为 `up` 之后把 `eth0` 和 `eth1` 放到 `bond0`。最后还可以指定一个接口停止之前的命令。这次还使用 `ifenslave`，通过发出 `-d` 选项在 `bond0` 调整为 `down` 之后把 `eth0` 和 `eth1` 设备从 `bond0` 分开，命令如下：`pre-down ifenslave -d bond0 eth0 eth1`。

最后，与 Red Hat 里一样，需要加载内核模块使设备启动。我们发出下面的命令。

```
$ sudo modprobe bonding mode=0 miimon=200
$ sudo /etc/init.d/networking restart
```

我们同样要重启主机。那么在发出下面的命令时，可以看到下面的输出。

```
$ sudo /sbin/ip addr show
2: eth0: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> ➡
mtu 1500 qdisc pfifo_fast master bond0 qlen 1000
    link/ether 00:0c:29:94:bd:33 brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> ➡
mtu 1500 qdisc pfifo_fast master bond0 qlen 1000
    link/ether 00:0c:29:94:bd:33 brd ff:ff:ff:ff:ff:ff
7: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> ➡
mtu 1500 qdisc noqueue
    link/ether 00:0c:29:94:bd:33 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.10/24 brd 192.168.0.255 scope global bond0==
    inet6 fe80::20c:29ff:fe94:bd33/64 scope link
        valid_lft forever preferred_lft forever
```

现在可以看到 IP 地址 192.168.0.10 赋给了 `bond0`，`eth0` 和 `eth1` 两者是 `bond0` 的从设备：`<BROADCAST, MULTICAST, SLAVE, UP, LOWER_UP>` `mtu 1500 qdisc pfifo_fast master bond0`。你还注意到 3 个设备都有同一个 MAC 地址 `00:0c:29:94:bd:33`，意思就是它们都可以响应对那个 MAC 地址的 ARP（Address Resolution Protocol，地址解析协议）请求。“TCP/IP

101”一节会深入解释 ARP。

本章已多次使用 ip 命令；现在来讲述如何使用它更改网络配置。

3. 使用 iptools2 测试网络配置

传统上，在 Linux 主机上可以使用 ifconfig 命令管理主机的网络接口，使用 route 命令管理路由表。iptools2 软件包提供的 ip 命令可用于这两项任务。它在管理接口和路由，特别在测试、添加和删除网络设置方面很有用。

可以使用 ip 命令向接口添加 IP 地址、移除 IP 地址，以及使接口状态为 up 或 down。你还可以使用它配置网络路由和路由表。

使用 ip 工具所做的配置改动不能持续到主机重启。要做更长久的改动，需使用 GUI 工具（system-config-network 或者 network-admin），或者直接编辑相关主机的网络配置文件，如本章早前所讲述的那样。

那么为什么需要 ip 命令呢？有时不想往接口永久添加一个 IP 地址，或者想在某个脚本运行时临时增加一个路由。使用这个工具可以轻松完成这两件事。

让我们分析一下 ip 命令。ip 命令的基本语法如下所示。

```
ip [ OPTIONS ] OBJECT { COMMAND | help }
```

ip 命令最重要的部分就是那些可操作的对象。最常见的对象是 link、address 和 route。表 6-3 描述了这三个对象以及该命令的其余对象。

表 6-3 ip 命令里可描述的对象

对象	说明
link	网络设备
address	接口的地址（IPv4 或 IPv6）
neighbour	ARP 或者 NDISC 的缓存输入项
route	路由表输入项
rule	路由策略数据库中的规则
maddress	组播地址
mroute	多播路由缓存输入项
tunnel	IP 上的隧道

使用 ip 命令可操作的对象，如接口和路由表，可以管理许多网络需求。ip 命令语法中指定的 COMMAND 是在那些对象上执行的动作。每个对象类型有它自己的一套合法的可执行命令。

让我们以对象 link 为例。你可以使用 up 或 down 命令设置连接的状态。

```
$ sudo /sbin/ip link set eth0 down
```

这个命令把 eth0 设备设置为（link set eth0）down 状态。link 就是接口，该命令会把接口状态设置为 down，很像早前所述的 ifconfig eth0 down 命令。下面给出 ip link show 命令（显示连接处于 down 状态）的输出。


```
$ sudo /sbin/ip link show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:0c:29:94:bd:3d brd ff:ff:ff:ff:ff:ff
    inet6 fe80::20c:29ff:fe94:bd3d/64 scope link
        valid_lft forever preferred_lft forever
```

通过<BROADCAST, MULTICAST, UP, LOWER_UP>可以看到 eth0 在本例中的状态为 up。通过下面的命令使它回到 down 的状态，然后再看一下它的状态。

```
$ sudo /sbin/ip link set down eth0
$ sudo /sbin/ip link show eth0
2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:0c:29:94:bd:3d brd ff:ff:ff:ff:ff:ff...
```

现在接口处于 down 状态，不能用它通信。可以看到，两个状态都不显示接口上的 IP 地址。现在就给 eth0 接口赋予一个地址。首先，使用/sbin/ip link set up eth0 使接口回到 up 状态（尽管不必设置接口的状态为 up）。然后给设备 eth0 添加 IP 地址 192.168.0.120/24。

```
$ sudo /sbin/ip addr add 192.168.0.120/24 dev eth0
```

通过使用 ip 命令显示接口信息，你可以看到地址 192.168.0.120 现在已经赋给以太网接口 eth0。/24 提供 255.255.255.0 的子网掩码。如果愿意的话，你可以选择/27、.224、/29 或.248。

```
$ sudo /sbin/ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:0c:29:94:bd:3d brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.120/24 scope global eth0
    inet6 fe80::20c:29ff:fe94:bd3d/64 scope link
        valid_lft forever preferred_lft forever
```

■注：即将看到的“Ping!”一节中将说明如何使用 ping 命令测试接口是否在工作。

下面删除接口的地址。为此，使用带 addr 对象和 del 命令的 ip 命令，如下所示。

```
$ sudo /sbin/ip addr del 192.168.0.120/24 dev eth0
```

在删除接口时必须记得添加正确的网络掩码；否则就会接收到一个错误信息。现在列出接口，我们可以看到该地址已经不再显示了。

```
3: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:0c:29:94:bd:3d brd ff:ff:ff:ff:ff:ff
    inet6 fe80::20c:29ff:fe94:bd3d/64 scope link
        valid_lft forever preferred_lft forever
```

■注：你可能觉得奇怪的是，使用 ip 命令配置的 IP 地址在运行 ifconfig 命令时竟不会出现。你需要使用 ip 命令查看它所创建的 IP 地址。

补充知识：ETHTOOL

ethtool 命令用于深入调查和处理接口。例如，你可以改变接口链路的速度或者双工设置。你可以通过发出下面的命令查看设备当前所使用的设置。

```
$ sudo ethtool eth0
Settings for eth0:
    Supported ports: [ MII ]
```



```

Supported link modes: 10baseT/Half 10baseT/Full
                      100baseT/Half 100baseT/Full
                      1000baseT/Full
Supports auto-negotiation: Yes
Advertised link modes: 10baseT/Half 10baseT/Full
                      100baseT/Half 100baseT/Full
                      1000baseT/Full
Advertised auto-negotiation: Yes
Speed: 100Mb/s
Duplex: Full
Port: MII
PHYAD: 0
Transceiver: external
Auto-negotiation: on
Supports Wake-on: g
Wake-on: d
Link detected: yes

```

它显示了 eth0 设备的当前设置。如果想改变双工模式和速度，要发出像下面这样的命令。

```
$ sudo ethtool eth0 -s speed 1000 duplex half
```

这里把以太网卡的速度改为 1000Mbit/s，双工模式设置为半双工。也可以使用其他设置。为了更好地理解它的可用设置，请阅读 ethtool 的 man 页。

6.1.4 添加路由与转发数据包

可以用 ip 命令完成的另一件常见的事情是向主机添加路由。路由就是主机为了访问别的网络或别的主机应该走的路径。

补充知识：IP 转发

Linux 主机一般默认关闭 IP 转发功能。IP 转发允许主机担当路由器的角色，从而可以把网络上的主机发给它的数据包导向目的主机。如果主机 IP 转发功能打开了，主机会从一个接口到另一个接口或从一个网络到另一个网络地转发数据包。这是一个内核级别的功能，你可以通过编辑/etc/sysctl.conf 文件打开或关闭它。这个文件允许在运行时配置内核参数，其中有许多其他参数都可以设置，包括内存、交换区和其他运行时可以调整的网络相关的内核参数。

内核把这类信息存储在 /proc 目录里。例如，IP 转发设置位于 /proc/sys/net/ipv4/ip_forward 文件中。内核在启动时创建这个文件系统，因此对它的直接改动不会持续到下次重启。如果它被设置为 0，那么 IP 转发功能就关闭了，主机就不会传递任何目的地不是自身的数据包。如果它被设置为 1，那么 IP 转发会把那些目的地不是自身的数据包重新导向目的主机。

为了使设置能够持续到下次主机重启，我们要去掉/etc/sysctl.conf 中下面一行的注释符号，保存文件，然后发出 sysctl -p 命令把改动加载到内核里。

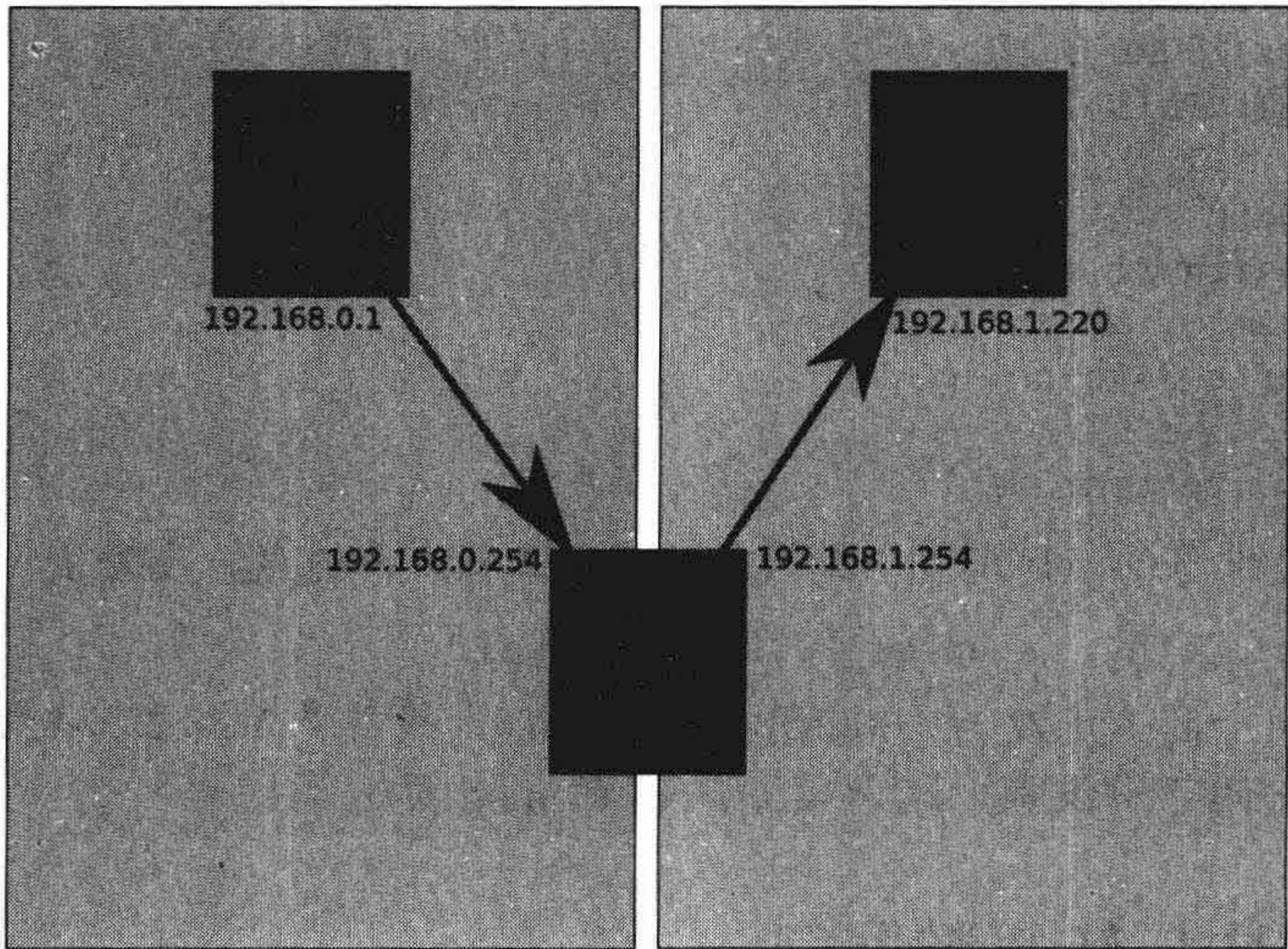
```
# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1
```

我们还可以通过回送数字 1 到文件 /proc/sys/net/ipv4/ip_forward 中立刻打开包转发功能，如下所示。

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

这个改动不能持续到重启，不过编辑/etc/sysctl 会使之长久。要了解更多 sysctl.conf 以及 sysctl 相关的使用信息，请阅读它们的 man 页。

我们用一个例子来说明如何使用 ip 命令添加一个路由。首先说明要达到的目标。我们工作于 IP 地址为 192.168.0.1 的主服务器主机——图 6-18 中的主机 A。网络中的防火墙/路由器的接口 eth0 上的 IP 地址为 192.168.0.254——主机 B。防火墙主机上的接口 eth0 还有一个 192.168.1.254/24 的地址。网络 192.168.0.1/24 是无线网络，把它分到它自己的子网。希望与之通信的主机 IP 地址为 192.168.1.220——主机 C。这也通过图 6-18 表示清楚了。



HOST A 主机 A; HOST B 主机 B; Host C 主机 C。

图 6-18 添加一个路由

主机 A 和主机 C 被适当地配置以便能够在网络之间路由。在它们的路由表中，它们把所有目的地为其他网络的请求通过它们的默认网关发送出去。在本例中，它们的默认网关是防火墙/路由器，默认网关的地址分别为 192.168.0.254 和 192.168.1.254。主机 A 不知道主机 B 的 eth0 接口有 192.168.1.0/24 的网络，也不知道主机 C 知道主机 B 有 192.168.0.0/24 的网络。

让我们看一看主机 A 的路由表。通过发出下面的命令，可以列出主机 A 的路由表。

```
$ sudo /sbin/ip route show
192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.1
169.254.0.0/16 dev eth0 scope link
127.0.0.0/8 dev lo scope link
default via 192.168.0.254 dev eth0 scope link
```

路由表显示了该主机所知道的网络。它知道它自己的网络，192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.1 显示了这一点。网络 169.254.0.0/16 dev eth0 scope link 是所谓的本地链路网络。它使主机无需复杂的网络服务（比如 DHCP 和 DNS）就可互相通信。

在路由输出的下一行 127.0.0.0/8 dev lo scope link 是有关回送地址网络的。许多服务使用本网络范围内的地址进行进程间通信，这个路由是回送设备 dev lo 上的。

最后看到的是默认路由 default via 192.168.0.254 dev eth0 scope link。默认路由所指向的主机可处理它自己网络上的主机路由请求。这些设备通常带一个以上接口或 IP 地址网络，因此可以沿着路由路径把路由请求传递给更远的主机，一直到最终的目的地。

让我们拿它和主机 B 的路由表做个比较。主机 B 是网络路由器，因此它的接口上应该有

一个以上的网络。

```
$ sudo /sbin/ip route show
10.204.2.10 dev ppp0 proto kernel scope link src 10.0.2.155
192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.254
192.168.1.0/24 dev eth0 proto kernel scope link src 192.168.1.254
169.254.0.0/16 dev eth0 scope link
default dev ppp0 scope link
```

首先让我们看看主机 B 上的 default 路由。默认路由显示，主机 B 会把它所不知道的网络路由请求发送给 dev ppp0。设备 ppp0 是与 ISP 的 PPP 连接。第一行和那个路由有关。它显示通过设备 ppp0 和 link src 地址 10.0.2.155 到达网络 10.204.2.10。10.204.2.10 主机在 ISP 那儿，并在做 PPP 连接时由 ISP 分配给我们。

我们知道主机 A 和主机 C 在不同的网络上，靠它们自己不能到达对方。为了做一个简单的连接，比如一个从主机 A 到主机 C 的 ping 连接，TCP/IP 数据包必须经过主机 B。ping 命令给命令行上指定的主机发送一个 ICMP 回送请求。回送请求是一个已定义的协议，它用计算机的语言对发出请求的主机说“我在这里”。

■注：Internet 控制消息协议（Internet Control Message Protocol, ICMP）是一种用于在主机间发送消息（一般为错误消息）的 TCP/IP 协议；但是在 ping 命令的情况中，它还可以发送一个报告型的回送答复。

那么让我们看看通过在主机 A 上发出 ping 命令能不能从主机 A 到达主机 C。（本章“Ping!”一节将更深入地解释 ping 命令。）

```
$ ping -c 4 192.168.1.220
PING 192.168.1.220 (192.168.1.220) 56(84) bytes of data.
64 bytes from 192.168.1.220: icmp_seq=1 ttl=64 time=1.79 ms
```

这里使用了 ping 命令（一种发送网络回音的方式），并用 -c 4 选项限制回音答复的次数为 4。这足以确认可以从主机 A 到达主机 C。这相对容易。让我们看一个不同的情况，在这种情况下必须添加自己的路由。从图 6-19 可以看到它的示意图。

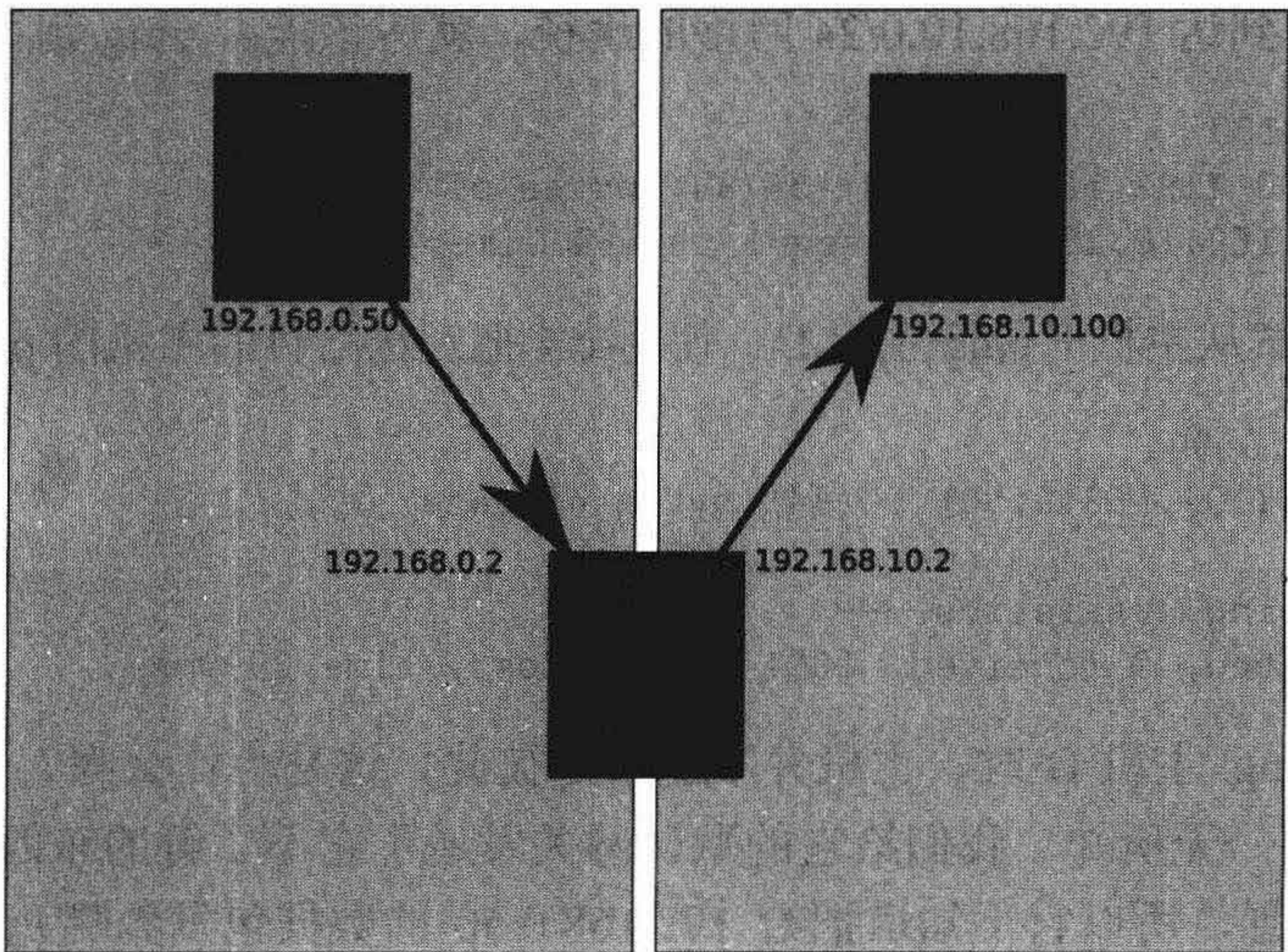


图 6-19 添加一个路由

在这种情况下，假定主机 D 试图通过主机 E 到达主机 F。主机 E 有两个以太网设备 `eth0` 和 `eth1`。设备 `eth0` 用 IP 地址 192.168.0.2 连在 192.168.0.0/24 网络上，`eth1` 用 192.168.10.2 的 IP 地址连在 192.168.10.0/24 网络上。主机 D 在 192.168.0.0/24 网络上，IP 地址为 192.168.0.50。主机 D 有一个默认路由 192.168.0.254。主机 F 在 192.168.10.0/24 网络上，IP 地址为 192.168.10.100。主机 F 有默认路由 192.168.10.254。

让我们看看它们的路由表。

```
Host D
192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.50
169.254.0.0/16 dev eth0 scope link
127.0.0.0/8 dev lo scope link
default via 192.168.0.254 dev eth0 scope link

Host E
192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.2
192.168.10.0/24 dev eth1 proto kernel scope link src 192.168.10.2
169.254.0.0/16 dev eth0 scope link
127.0.0.0/8 dev lo scope link
default via 192.168.0.254 dev eth0 scope link

Host F
192.168.10.0/24 dev eth0 proto kernel scope link src 192.168.10.100
169.254.0.0/16 dev eth0 scope link
127.0.0.0/8 dev lo scope link
default via 192.168.10.254 dev eth0 scope link
```

你可以看到主机 E 包含两个接口，并与两个不同的网络相连。为了让主机 D 看到主机 F，要告诉主机 D 通过主机 E 到达主机 F。为此，我们添加一个如下的路由。

```
$ sudo /sbin/ip route add 192.168.10.0/24 via 192.168.0.2 dev eth0
```

在这里不得不结合使用 `route` 对象和 `ip` 命令从主机 D 向 192.168.10.0/24 网络添加一个路由。把这台主机的 IP 地址 192.168.0.2（它是我们网络上到达目标网络的路由器）提供给 `ip` 命令。

既然主机 D 有到达 192.168.10.0/24 网络的途径，就应该能够 `ping` 到主机 F。

```
$ ping 192.168.10.100
PING 192.168.10.100 (192.168.10.100) 56(84) bytes of data.
64 bytes from 192.168.10.100: icmp_seq=1 ttl=64 time=1.24 ms
```

现在让我们尝试从主机 F `ping` 主机 D。使用 `-c 4` 把发送给目的主机的 `ping` 次数限制为 4。

```
ping -c 4 192.168.0.50
PING 192.168.0.50 (192.168.0.50) 56(84) bytes of data.

--- 192.168.0.50 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 2999ms
```

当从主机 F `ping` 主机 D 时，主机并不能看到彼此。这是为什么呢？这是因为主机 F 对 192.168.0.0/24 网络一无所知。我们给它的默认网关发送了它不了解的东西，并且因为主机 F 的默认网关没有配置，所以它不知道把给 192.168.0.50 的数据包发到哪儿，它会把这些包发送给它的默认网关。最后这些包会消失。

■注：TCP/IP 协议的一部分是 TTL——time to live——经过一段时间后，数据包会被上行路由器忽略并丢掉。TTL 用一个数字表示，现在一般是 64，但可以是 32 或 128。数据包每通过一个路由器（或一跳），它们的 TTL 值就减 1。如果在它到达某个路由器时 TTL 值为 1 并且那个路由器不是最后的目的地，这个路由器就会丢掉数据包并使用 ICMP 协议发送一个“TTL 传输过期”的消息。你很快就会在“MTR”一节看到它的用处。

那么如何解决这个问题？好吧，往主机 F 添加一个路由，告诉它如何通过主机 E 把包发送给主机 D。

```
$ sudo /sbin/ip route add 192.168.0.0/24 via 192.168.10.2 dev eth0
```

现在如果主机 F 尝试 ping 主机 D，就会得到下面的结果。

```
$ ping 192.168.0.50
PING 192.168.0.50 (192.168.0.50) 56(84) bytes of data.
64 bytes from 192.168.0.50: icmp_seq=1 ttl=64 time=1.24 ms
```

往主机 F 添加路由之后，马上就可以到达主机 D。把所有这些信息结合起来，可以在脚本里使用这些命令，使接口启动，添加 IP 地址，并添加路由。

```
#!/bin/bash

# bring up the interface
ip link set eth1 up

# add an address
ip addr add 192.168.10.1/24 dev eth1

# add a route
ip route add 192.168.100.1 via 192.168.10.254 dev eth1

# ping the host
ping 192.168.100.1 -c 4

# bring down the route, remove the address and bring down the interface.
ip route del 192.168.100.1 via 192.168.10.254 dev eth1
ip addr del 192.168.10.1/24 dev eth1
ip link set eth1 down

exit 0
```

我们使用了同一个命令来控制接口，使它启动，添加地址，然后添加路由。接下来再用同样的命令使它停止。这种脚本可用来从一台主机到另一台主机复制文件或者做任何类型的连接。

现在让我们进一步深入探究如何使用几个简单的实用工具测试连接并诊断网络故障。

6.2 一般网络故障检修

网络可能会出问题。有时不能连接到服务，或者网络的某部分配置不正确。为了识别并

解决这些问题，需要一套工具。我们将分析下述多种工具。

- **ping**: 一个在主机之间发送包以确认连接的命令。
- **tcpdump**: 一个显示和捕获网络流量的命令。
- **mtr**: 一个网络诊断工具。
- **nc**: netcat, 另一个网络诊断和测试工具。

6.2.1 Ping!

可能人们检查网络最常使用的工具就是叫作 **ping** 的命令。你可以 **ping** 一台主机或一个接口看它是否回应。在对 **ping** 命令的输出回应中，会显示那个回应所花的时间。你可以 **ping** 世界的另一端，在几毫秒内就可接收到回应。这不错。如果得到回应的时间是秒级的时间或者得到一个“**host unreachable**”的消息，那就很糟。当然，如果正在 **ping** 的主机在别的大陆或者是通过卫星链路连接的，**ping** 的时间将会比 **ping** 眼前桌下的主机要长。

正好使用 **ping** 命令测试我们的路由器，如下所示。

```
$ ping 192.168.0.50
PING 192.168.0.50 (192.168.0.50) 56(84) bytes of data.
64 bytes from 192.168.0.50: icmp_seq=1 ttl=64 time=1.24 ms
```

ping 命令可带下面的参数。对于其他参数，请参阅它的 **man** 页。

```
ping -c <count> -i <interval> -I <interface address> -s <packet size> destination
```

ping 命令会不确定地向某个主机发送查验信息，除非停止它（通常使用 **Ctrl+C**），或者使用 **-c number** 选项提供尝试 **ping** 的次数。

你还可以用 **-i number** 表明 **ping** 之间的时间间隔（单位为秒），用 **-I IP address** 来指定希望用作源地址的接口。这便于在拥有多个接口时测试其中的一个。如果没有定义地址，它通常会使用主机上的第一接口（通常为 **eth0**）来进行 **ping**。你还可以使用 **-s number of bytes** 指定包的大小，它在测试带宽问题或 **MTU** 设置问题方面有用。

■注：如本章早前所提及的那样，**MTU** 是最大传输单元。它用于把数据包数量限制在那个网络上的网络设备可以处理的范围。它一般设置为 1500 字节，但对于特大帧它可以达 9000 字节。大部分 Internet 设备的 **MTU** 是 1472。

为了测试网络，可做的首要事情之一是使用 **ping** 命令 **ping** 已配置的接口。如果它们对来自主机的 **ping** 有回应，它们就是活动的。它们可能配置不适当，但至少在工作。可以如下这样做。

```
$ ping 192.168.0.253
PING 192.168.0.253 (192.168.0.253) 56(84) bytes of data.
64 bytes from 192.168.0.253: icmp_seq=2 ttl=128 time=1.68 ms
```

这里给主机的本地 **IP** 地址发送了一个查验信息，可以看到能证明一条连接存在的一连串的回应和所花费的时间。如果没有接收到回应，就知道接口或网络出错了。

下一步是 **ping** 自己网络上的其他主机，如默认网关或 **DNS** 服务器（这两台主机对你的

Internet 通信很关键)。

```
$ ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=128 time=2.97 ms
```

如果可以到达那里，就知道主机可以到达其他主机。然后可以 ping 自己网络外的一个主机，比方说 www.ibm.com 或者 www.google.com，然后测试它的回应。

```
$ ping www.google.com
PING www.l.google.com (150.101.98.222) 56(84) bytes of data.
64 bytes from g222.internode.on.net (150.101.98.222): icmp_seq=1 ttl=59 time=20.0 ms
```

如果在连接 Internet 上的主机时遇到了问题，那么可能有许多原因。有可能是局部 Internet 因为某个核心路由器损坏而中断了。在这种情况下，对自己网络的所有 ping 命令都会有回应，但对 Internet 上的某些主机可能 ping 不通。如果不能从 Internet 上的一台主机得到回应，那么尝试另一台主机看看问题是不是出在本地或者这条线路上的其他地方。一些远程网络会主动阻止 ICMP 流量。这些主机不会回应 ping 命令，因此不要恐慌，要先检查一下其他主机的回应。在这些情形下，结合其他工具进行调查也很便利，其中的一个工具就是 mtr。

6.2.2 MTR

如果可以 ping 自己网络内的主机但不能 ping 网络外的主机，可以使用像 traceroute 或 mtr 这样的工具来诊断。两者提供类似的服务：它们记录用来到达目的主机的路由。两者也使用“TTL 传输过期”消息来记录沿线的那些主机。这些消息是什么意思？对，正如我们所说的那样，TTL 用来杀掉 TCP/IP 包，因此它们不会像大型强子对撞机里某些损失粒子一样永远在 Internet 上流动。有礼貌的路由器在杀掉数据包时会发回“TTL 传输过期”的消息，这些命令正是利用了这一点。

让我们看一个例子。如果 ping www.ibm.com 并设置 TTL 为 1，它就会到达 www.ibm.com 所解析途径上的第一个路由器。那个路由器看到 TTL 是 1，就立刻丢掉这个包并发送传输过期消息。

```
$ ping -t 1 -c 1 www.ibm.com
PING www.ibm.com.cs186.net (129.42.60.216) 56(84) bytes of data.
From 192.168.0.254 (192.168.0.254) icmp_seq=1 Time to live exceeded

--- www.ibm.com.cs186.net ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms
```

mtr 和 traceroute 程序利用发回的包找到路由器的 IP 地址（因为包含回复的 TCP/IP 包会保留此信息）。然后它显示这个信息并发出下一个 ping 命令，这次 TTL 设置为 2。一旦 ping 到达目的地，我们就应该能接收到标准的回送回应。有些路由器配置为不发送这些 ICMP 消息，因此在记录的输出处看到的是空白。

看一看 mtr 的输出。在这里使用 mtr 记录主机 au-mel-rhel-1 和 www.ibm.com 之间的路由，使用的命令如下所示。


```
$ sudo /usr/sbin/mtr www.ibm.com
My traceroute [v0.71]
au-mel-rhel-1 (0.0.0.0) Thu Jan 8 17:38:23 2009
Keys: Help Display mode Restart statistics Order of fields quit
Packets Pings

Host Loss% Last Avg Best Wrst StDev
1. 192.168.0.254 0.0% 0.6 1.0 0.0 1.9 0.5 2.0
2. loop0.lns10.mel6.domain.com 0.0% 9.1 8.5 7.9 9.1 0.5
3. vlan13.cor2.mel6.domain.com 0.0% 9.2 9.7 8.0 14.5 2.4
4. gi0-0-4.bdr1.mel6.domain.com 0.0% 201.7 202.0 201.0 202.9 0.7
5. pos2-3.bdr1.syd6.domain.com 0.0% 168.2 169.1 168.2 169.5 0.5
6. pos5-0.bdr1.lax1.domain.com 0.0% 201.1 200.8 200.1 201.5 0.6
7. Gb3-2.GW1.LAX15.ALTER.NET 0.0% 169.1 168.4 167.7 169.1 0.6
8. 01.XL3.LAX15.ALTER.NET 0.0% 168.0 169.1 167.6 172.2 1.9
9. 02.XT3.STL3.ALTER.NET 0.0% 214.2 218.8 213.9 236.3 9.8
10. POS6-0.GW8.STL3.ALTER.NET 0.0% 247.7 247.3 246.7 247.7 0.4
11. ibm-gw.customer.alter.net 0.0% 216.5 216.4 215.5 217.0 0.5
12. ???
```

该命令的输出显示，第一跳（数据包经过的每个主机/路由器都叫一跳）是我们的防火墙路由器。接下来是 ISP 所提供的 Internet 连接默认网关。随着沿途经过每一跳，每个路由设备的信息就被记录下来了。在倒数第二跳到达 IBM 网关。在最后一跳，看来 IBM 拒绝从它的网络内部返回 ICMP 包，mtr 打印出???, 因为它没有接收到“TTL 传输过期”的消息。

6.2.3 TCP/IP 101

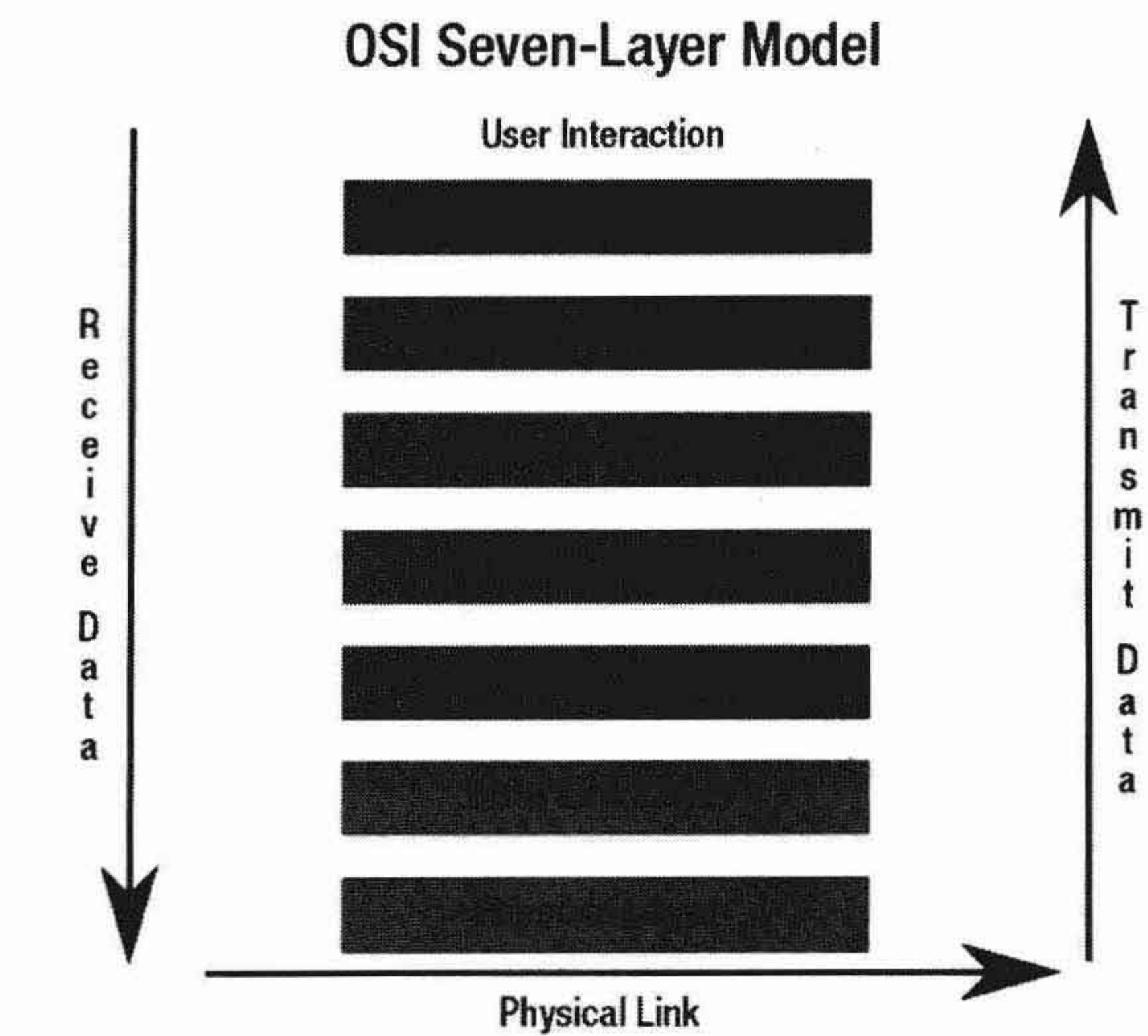
该深入地研究 TCP/IP 了。你可能熟悉 IP 地址，但它是如何与 TCP/IP 其余部分配合的呢？IP 地址用来寻找网络上其他主机以及其他主机找你。但这是如何做到的呢？

你一定有兴趣知道，当发起一个 TCP/IP 连接（也叫套接字）时，一个 3 阶段的处理过程使此连接进入一个“建立”状态，意思就是两台主机都意识到了它们彼此之间的套接字，同意它们即将发送数据的方式，并且已准备好发送那些数据。第一个阶段是主机通过给希望与之通信的主机发送一个叫作 SYN 的包发起套接字。那台主机用另一个包，就是有名的 SYN, ACK 包来回应，表明它为通信的开始做好了准备。然后发起端主机发送另一个 SYN, ACK 包，确认收到数据包并告诉远程主机它即将开始发送数据。当它们完成了通信，会有一个 FIN 包来关闭连接。这是对 TCP/IP 通信过程的一个基本的概述。

■注：要更深入研究实际的包是什么样的，请到这里 http://www.tcpipguide.com/free/t_TCPMessageSegmentFormat-3.htm 阅读有关 TCP 部分的讨论。还可以到下面的网址看一个简易的袖珍指南：<http://www.sans.org/resources/tcpip.pdf?ref=3871>。

它所依赖的协议由 7 层构成。每一层在线路上的主机之间的数据通信过程中都有特定的职责。在网络诊断方面，一般关注第 1、2 和 3 层。从图 6-20 可以看到对这些层的描述。

■注：有关其余层更深入的讨论以及 TCP/IP 的更多信息，请查看下面的网址：http://catalyst.washington.edu/help/computing_fundamentals/networking/osi.html。



OSI Seven-Layer Model OSI 七层模型
Receive Data 接收数据 Transmit Data 发送数据
User Interaction 用户交互
Application Layer 应用层
Presentation Layer 表示层
Session Layer 会话层
Transport Layer 传输层
Network Layer 网络层
Data Link Layer 数据链路层
Physical Layer 物理层
Physical Link 物理链路

图 6-20 OSI 层模型

第一层是物理层，如图 6-20 底端出现的物理层所示。在诊断网络问题时，轻摇主机与网络其余部分的连线绝不会造成伤害。另外，如果其他尝试都失败了，你可以尝试更换网线。如果网线有故障，与主机连接着的交换机/集线器和网卡的灯可能不亮。如果这些灯没有亮，主机将不能与其他主机通信，因此首先尝试更换网线，然后更换它所连接的交换机上的端口，最后换网卡。

第二层数据链路层提供线路上实际的通信协议。这层很少发生问题。这里正是用 ARP 匹配 IP 地址与 MAC 地址的地方。当网络上的两个主机有相同的 IP 地址而有不同的 MAC 地址时，主机也许会试图往错误的主机发送数据。在这种情况下，ARP 表就需要刷新，IP 地址不正确的主机要离线。

第三层是网络层。它能够发现到达目的地的路由并发送数据，并在这个过程中检查错误。这是 IP 层，因此还可以负责 IPsec 隧道。另外，这一层还负责回应 ping 和其他路由请求。

6.2.4 tcpdump 命令

在第一层不能轻易地查看通信过程，但在第二、三层使用数据包探查（packet-sniffing）

软件可以查看。查看通信详情的程序中有一个是 `tcpdump` 命令行工具。使用 `tcpdump` 命令以及与其类似的那些工具可以查看线路上数据包层面的流量。你可以看到数据包进出于主机。`tcpdump` 命令在不带任何表达式运行时打印经过接口的每一个数据包。使用表达式可以缩减要显示的数据包类型数量。更多信息，可到 `tcpdump` 的 man 页查看。

■注：你也可以尝试另一个叫 Wireshark 的程序。它有一个非常好的 GUI，使用它可以轻松筛选流量。还有一个命令行实用工具叫 `tshark`，它的操作方式类似于 `tcpdump`。有关 Wireshark 的更多信息，可以查阅 <http://www.wireshark.org/>。

当使用 `tcpdump` 时，输出结果如下所示。

```
$ sudo /usr/sbin/tcpdump -i eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
16:58:35.901825 arp who-has 192.168.0.254 tell 192.168.0.1
16:58:35.902941 arp who-has 192.168.0.254 tell 192.168.0.1
16:58:35.903145 arp reply 192.168.0.254 is-at 00:0b:cd:49:c0:7f (oui Unknown)
16:58:35.904845 IP 192.168.0.254 > 192.168.0.1.smtp: S 2737972828:2737972828(0) ➡
win 16384 <mss 1460,nop,wscale 0,nop,nop,sackOK>
16:58:35.917231 IP 192.168.0.1.smtp > 192.168.0.254: S 4235714640:4235714640(0) ➡
ack 2737972829 win 5840 <mss 460,nop,nop,sackOK,nop,wscale 4>
16:58:35.907176 IP 192.168.0.254 > 192.168.0.1.smtp: . ack 1 win 17520
16:58:35.922434 IP 192.168.0.1.smtp > 192.168.0.254: P 1:50(49) ack 1 win 365
16:58:36.101483 IP 192.168.0.254 > 192.168.0.1.smtp: . ack 50 win 17471
```

我们发出了这个命令，告诉它转存所有通过 `eth0` 接口的流量，并使用 `-i` 选项指定要分析的那个接口。对于输出，可以像下面这样分解这些行。

timestamp source > destination : flags

上述输出的头两行信息说明这个命令在做什么，本例是 `listening on eth0`。其余行是线路上真正的数据包。每一行的第一组数据是时间戳，比如 `16:58:35.901825`。

`arp who-has 192.168.0.254 tell 192.168.0.1`

第一个字段（拿掉时间戳后）是 TCP/IP 模型里的协议。这是一个 ARP 请求，ARP 在 TCP/IP 协议栈的第二层（或者数据链路层）工作。ARP 用来匹配 MAC 地址与 IP 地址。可以看出 192.168.0.1 想知道谁有 192.168.0.254 的地址。有一个 ARP 回复说 192.168.0.254 所在的 MAC 地址是 `00:0b:cd:49:c0:7f`。

既然 192.168.0.1 知道发送它的包的位置，那么就通过发送一个 SYN 包尝试建立一个套接字。SYN 包携带置位的 SYN 比特，并且有一个初始序列号码 `S 2737972828:2737972828(0)`。

```
IP 192.168.0.254 > 192.168.0.1.smtp: S 2737972828:2737972828(0) ➡
win 16384 <mss 1460,nop,wscale 0,nop,nop,sackOK>
```

源主机和目的主机被描述为 `192.168.0.254 > 192.168.0.1.smtp`，这里的 `.smtp` 是正要连接的端口。它映射为端口 25。这是一条正在建立的到 SMTP 邮件服务器的连接。

让我们看看接下来的部分：`S 2737972828:2737972828(0)`。源主机和目的主机之后的 `S` 表明这是一个 SYN 请求，当前正在建立一个连接。接着就是包的初始序列号码

2737972828:2737972828(0)。序列号随机生成，用来排序和匹配包。序列号之后的 (0) 意思是这是一个 0 字节的包（就是说它不包含有效载荷）。

其他标志 win 16384 <mss 1460, nop, wscale 0, nop, nop, sackOK> 提供通信中的其他信息，如滑动窗口大小、最大段大小等。

■注：更多信息见 http://www.tcpipguide.com/free/t_TCPMaximumSegmentSizeMSSandRelationshiptoIPDatagra.htm。

接下来的包是来自 192.168.0.1 的回复。

```
IP 192.168.0.1.smtp > 192.168.0.254: S 4235714640:4235714640(0) ➡
ack 2737972829 win 5840 <mss 460,nop,nop,sackOK,nop,wscale 4>
```

这个包的序列号为 S4235714640: 4235714640 (0)。这是另一个随机生成的号，数据载荷还是 0。附在序列号后面的是一个 ACK 回应 ack 2737972829。这是原来的初始序列号加 1，表明它承认了第一个序列号。

紧接着的这个包是另外一个确认包，由发起主机发送。

```
IP 192.168.0.254 > 192.168.0.1.smtp: . ack 1 win 17520
```

在这个输出中，圆点 (.) 意思是没有标志被设置。ack 1 表明这是一个确认包，并且从现在开始，tcpdump 将显示当前序列和初始序列的差别。这是在两台主机之间建立连接所需的最后一次通信。

最后两个包是数据交换。

```
IP 192.168.0.1.smtp > 192.168.0.254: P 1:50(49) ack 1 win 365
P 192.168.0.254 > 192.168.0.1.smtp: . ack 50 win 17471
```

邮件服务器发送一条消息给 192.168.0.254 上的客户端，正如 P 1:50(49) ack 1 所预示的那样。它在载荷中推进了 49 字节数据 (P, 49)，并确认了上次通信。最后一次通信是 192.168.0.254 对上述包的确认：ack, 50。

那么既然已经了解了如何使用 tcpdump 这样的数据包探查程序在最基本的层面上查看通信过程，就让我们看一看另外一种有用的工具 netcat。

■注：如果对 tcpdump 和建立连接的更深入讨论有兴趣，可以试着看看这篇文章：<http://www.linuxjournal.com/article/6447>。

6.2.5 Netcat 工具

另一种在诊断网络问题方面非常有用的工具是 nc，或称 netcat 命令。使用这个工具可以测试到达其他主机和它们监听所用端口的能力。

当希望测试经过一个防火墙到一个端口的连接时，这个工具特别方便。让我们测试一下防火墙是否允许从主机 192.168.0.254 连接到主机 192.168.0.1 上的端口 80。

首先，在主机 192.168.0.1 上，要确保停止网络服务器。比如在 Red Hat 上，发出下面这

个命令。

```
$ sudo /sbin/service httpd stop
```

然后在 IP 地址为 192.168.0.1 的主机上启动 nc 命令，并使用 -l 或叫监听选项。

```
$ sudo nc -l 80
```

这样就通过这个端口把 nc 命令捆绑到所有接口上了。我们可以通过运行另一个叫作 netstat 的命令测试一下。

```
$ sudo netstat -lpt
tcp 0 0 *:http *:~ LISTEN 18618/nc .
```

发起的 netstat 命令带 3 个选项。-l 选项告诉 netstat 命令运行并监听网络连接。-p 选项告诉 netstat 显示正在使用这些连接的应用程序，最后一个选项 -t 告诉 netstat 只寻找 TCP 连接。

netstat 命令显示正在主机的某些端口上监听的程序。从上述输出中可以看到 PID 为 18618 的程序 nc 正在监听端口 80 上的 TCP 连接。*:http 表示它正在监听端口 80（:http 端口映射为端口 80）上所有可用的地址（网络接口的 IP 地址）。好，这样就知道 nc 命令正在监听和等待连接。接下来测试 IP 地址为 192.168.0.254 的主机的连接能力。

使用 telnet 命令创建一个到主机 192.168.0.1 端口 80 的连接，如下面的例子所示。

```
$ telnet 192.168.0.1 80
Trying 192.168.0.1...
Connected to 192.168.0.1.
Escape character is '^]'.
hello host
```

telnet 程序允许测试两台主机之间的连接并发送文本到远程主机。当在连接窗口键入文本并按下回车键时，可以看到键入的内容在 au-mel-rhel-1 主机上回显了。

```
$ sudo nc -l 80
hello host
```

现在知道主机可以连接到 au-mel-rhel-1 主机的端口 80，确认防火墙规则在起作用（或者太宽松了，如果正试图阻止端口 80 的话）。

6.2.6 dig 它?

dig 是另一种解析 DNS 问题的便利工具。如果结合其他如 ping 和 nc 那样的工具使用，就能解决许多问题。dig 命令是 domain information groper（域名信息探索者）的简写，它用来查询 DNS 服务器。这个命令用起来很简单，它通过查询可在 /etc/resolv.conf 文件里找到的 nameserver 来解析完整合格的域名。

/etc/resolv.conf 文件用来存储 nameserver 信息，因此主机知道要向哪台 DNS 服务器查询域名的解析结果。/etc/resolv.conf 文件看起来如下所示。

```
$ sudo cat /etc/resolv.conf
; generated by /sbin/dhclient-script
search example.com
```



```
nameserver 192.168.0.1
nameserver 192.168.0.254
```

首先可以看到 `resolv.conf` 文件由 DHCP 客户端产生。默认的 `search` 域是 `example.com`，任何主机名搜索都会在查询的末端追加这个域。接下来是希望查询其域名解析结果的 `nameserver(s)`。它们应该是 IP 地址的形式。如果第一个 `nameserver` 不可用，第二个会派上用场。

`dig` 命令会查询这些 `nameservers`，除非另外指明。让我们看一个查询。

```
$ dig www.google.com

; <<>> DiG 9.5.0-P2 <<>> www.google.com
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46826
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 7, ADDITIONAL: 7

;; QUESTION SECTION:
;www.google.com.IN A

;; ANSWER SECTION:
www.google.com. 209821 IN CNAME www.l.google.com.
www.l.google.com. 92 IN A 74.125.95.103
www.l.google.com. 92 IN A 74.125.95.104
www.l.google.com. 92 IN A 74.125.95.147
www.l.google.com. 92 IN A 74.125.95.99

;; AUTHORITY SECTION:
l.google.com. 13489 IN NS g.l.google.com.
l.google.com. 13489 IN NS a.l.google.com.
l.google.com. 13489 IN NS b.l.google.com.
l.google.com. 13489 IN NS c.l.google.com.
l.google.com. 13489 IN NS d.l.google.com.
l.google.com. 13489 IN NS e.l.google.com.
l.google.com. 13489 IN NS f.l.google.com.

;; ADDITIONAL SECTION:
a.l.google.com. 75270 IN A 209.85.139.9
b.l.google.com. 164012 IN A 74.125.45.9
c.l.google.com. 127883 IN A 64.233.161.9
d.l.google.com. 63971 IN A 74.125.77.9
e.l.google.com. 46712 IN A 209.85.137.9
f.l.google.com. 164012 IN A 72.14.235.9
g.l.google.com. 42069 IN A 74.125.95.9

;; Query time: 29 msec
;; SERVER: 64.5.53.6#53(64.5.53.6)
;; WHEN: Tue Apr 7 07:15:52 2009
;; MSG SIZE rcvd: 340
```

从 ANSWER SECTION 部分可以看到，`www.google.com` 主机名被解析为 DNS CNAME `www.l.google.com`，而它有 4 个可能的 IP 地址可作为最终解析的结果。AUTHORITY SECTION 部分说明哪一个 `nameservers` 负责提供 `www.google.com` 的信息。ADDITIONAL SECTION 部

分说明 AUTHORITY SECTION 的 nameservers 会解析成什么 IP 地址。

IN A 表示一个 Internet 地址（或相关记录）。CNAME 用来为主机起别名。IN NS 表示一个 nameserver 记录。上述代码里这三部分显示的数字如 92、13489 和 75270，表示记录会缓存多少秒。当这些数字为 0 时，nameserver 就会查询授权 DNS 服务器，看看它是否已更改。

在 dig 输出的底端可以看到提供回应的 SERVER 64.5.53.6 和查询所花费的时间 29 msec。

你可以使用 dig 和 @ 符号查询特定的 nameserver。你还可以使用 -t type 选项查询某些记录类型。例如，如果想测试 DNS 服务器是否正常工作，我们可以使用 dig 查询 Google 邮件服务器的 IP 地址。

```
$ dig @192.168.0.1 -t MX google.com

<snip>
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12164
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 4, ADDITIONAL: 4

;; QUESTION SECTION:
;google.com.      IN      MX

;; ANSWER SECTION:
google.com. 557 IN MX 10 smtp1.google.com.
<snip>
google.com. 557 IN MX 10 smtp4.google.com.

;; AUTHORITY SECTION:
google.com. 73806 IN NS ns1.google.com.
<snip>
google.com. 73806 IN NS ns4.google.com.

;; ADDITIONAL SECTION:
ns1.google.com. 36427 IN A 216.239.32.10
<snip>
ns4.google.com. 36427 IN A 216.239.38.10

;; Query time: 1 msec
;; SERVER: 192.168.0.1#53(192.168.0.1)
<snip>
```

我们可以使用这个信息比较其他 nameservers 的回应。如果 nameserver 提供的信息与之相同，DNS 服务器就在正确地工作。如果有差别或者没有返回结果，就必须深入调查 DNS 设置。第 9 章将详述 DNS 和 dig 命令。

6.2.7 其他故障诊断工具

刚探讨过的程序涉及了基本的网络故障诊断。许多其他网络工具可用来帮助进一步诊断问题。我们简要地介绍了 netstat 命令，但是还有其他大多数命令没有讲。表 6-4 列出了一些最普通的故障诊断命令和对它们的描述。

表 6-4 其他简易网络诊断工具	
工具	说明
netstat	此工具提供正在主机上监听的设备状态以及更多信息，包括路由和数据统计
host	又一种诊断 DNS 服务器的工具。第 9 章将说明如何使用 host
telnet	此工具允许与远程主机的端口进行 TCP 连接。可以测试对来自远程服务的命令的回应
openssl s_client	它对测试 SSL 连接有用，因为它可以尝试和远程主机上的端口建立 SSL 连接
arp	这个程序查询并管理 ARP 缓存表，并允许删除 ARP 输入项

表 6-4 中的有用工具列表绝不完整。不过，只用这几样工具就可以诊断许多一般的问题了。

6.3 Netfilter 与 iptables

现在许多人都理解什么是防火墙，因为甚至在最简单的主机上都必须安装防火墙。你可能已经熟悉运行在你台式机上的防火墙。这些简单的防火墙可用来阻止不想要的流量进出于单个主机，但它们也就能做这些事情了。

Netfilter 是一个综合的防火墙程序，它可以围绕一个网络或一台单独的主机工作。它不仅可以阻止不想要的流量以及路由网络周围的数据包，还可以给流量整形。“给流量整形？”你可能想，“那是什么？”

数据包整形（packet shaping）是对进出网络的数据包做修改的能力。利用它可以增加或减少某些连接的带宽，优化其他包的发送，或者确保特定类型的包的性能。这对那些希望确保 VoIP 通话不因有人下载一些疯狂青蛙手机铃声而中断的公司有好处。

6.4 Netfilter/iptables 是如何工作的？

iptables 命令是 Netfilter 的用户空间管理工具。Netfilter 是 Paul “Rusty” Russell 开创的，并且从版本 2.4 开始就已经进入 Linux 内核。它允许操作系统在内核层面上执行包过滤和整形，这使它处于比用户空间程序更少的限制下。这对专用防火墙和路由器主机特别有用。

■注：术语“用户空间程序”（user-space program）是指终端用户用来配置操作系统某部分的工具。在本例中，内部操作系统组件叫 Netfilter；用户空间组件，即用户配置 Netfilter 的命令，叫 iptables。

包过滤和整形是指在数据包进入或离开一个主机时按照一套标准或规则对包做改动或丢弃。Netfilter 通过在数据包进入、经过和/或离开主机时重写包头做到这一点。

Netfilter 是一个有状态的包过滤防火墙。防火墙有两种类型：有状态的和无状态的。一个无状态的包过滤防火墙只分析一个包的包头来寻找过滤信息。它孤立地看每一个包，因此没办法确定一个包是否为已有连接的一部分或者是否是一个孤立的恶意包。一个有状态的防

防火墙保留经过它的连接状态信息。这样就使防火墙可以基于连接状态过滤包，从而提供了对流量的相当细粒度的控制。

Netfilter 在用户空间的控制和配置由 iptables 命令来执行。在 Linux 内核以前的版本里，这个功能由别的命令提供。本章会经常使用 iptables 谈及一般的防火墙技术。大多数基于 Linux 的发行版都有 iptables 软件包，但它们也可能有自己的规则配置工具。

Netfilter 通过查阅一组表的方式工作。这些表包含链 (chains)，而链又包括单独的规则。链保存成组的类似规则；例如，一组管理输入流量的规则可能保存在一个链里。规则是基本的 Netfilter 配置项，它包括匹配特定流量的标准以及对匹配的流量执行的动作。

当前正被主机处理的流量与这些规则比对，如果数据包满足某个规则的挑选标准，那条规则所指定的动作，就是大家所知的目标 (target) 就被执行。其中，这些动作可能是忽略此包、接受此包、拒绝此包或者把此包传给别的规则以进行更细化的处理。让我们看一个例子。比如说网络服务器上的以太网接口刚接收到一个来自 Internet 的数据包。此包按照规则接受检查并与它们的挑选标准做对比。挑选标准包括如目的地 IP 和目的地端口这样的内容。比方说，希望 HTTP 端口 80 的网络输入流量进入网络服务器。如果进来的流量和这些标准匹配，就规定一个允许它通过的动作。

每一条 iptables 规则都依靠指定一组网络参数作为挑选标准来为本规则挑选数据包和流量。许多网络参数都可以用来构建 iptables 规则。比如说，两台主机之间的网络连接被称为套接字。它是源 IP 地址、源端口、目的地 IP 地址和目的地端口的组合。4 个参数在建立连接时都要有，iptables 可以使用这些值过滤输入主机和主机输出的流量。另外，如果看一看基于 TCP/IP 的网络通信执行过程，就可以看到 3 个最常使用的协议：ICMP、传输控制协议 (Transmission Control Protocol, TCP) 和用户数据报协议 (User Datagram Protocol, UDP)。iptables 防火墙可以轻易地把这些不同类型的协议与其他协议区分开。

在只有这 5 个参数 (源和目的 IP 地址、源和目的端口、协议类型) 的情况下，你可以立刻开始构建一些有用的过滤规则。但是在开始构建这些规则之前，要理解 iptables 规则是如何构造和相互影响的。而且为了能够理解这些，你需要更深入地理解一些原始的 iptables 概念，如表、链和策略。接下来就阐述这些概念，并涉及网络地址转换 (network address translation, NAT) 的概念。

6.4.1 表

我们讲过，Netfilter 有规则表，可以根据它对比网络流量，并可能采取某个行动。Netfilter 有 4 个可以保存流量处理规则的内置表。第一个是 filter 表，它是默认的表，用于所有与流量过滤相关的规则。第二个是 nat 表，它处理 NAT 规则。接下来是 mangle 表，它涉及各种包改动功能。最后一个 raw 表，它用来免除数据包的连接追踪，并且在其他任何 Netfilter 表之前先调用。

6.4.2 链

每一个 Netfilter 表，filter、nat、mangle 和 raw 都包含成组的预定义钩子 (hook)，Netfilter

会按次序处理它们。这些钩子包含排序的规则编组，即所谓的链。每个表都包含内置的默认链。表 6-5 描述了这些内置链。

链	说明
INPUT	用来为输入本主机接口的包安排规则顺序。只能在 filter 和 mangle 表中找到
FORWARD	用来为目的地是其他主机的包安排规则顺序。只能在 filter 和 mangle 表中找到
OUTPUT	用来为本主机接口输出的包安排规则顺序。可以在 filter、nat、mangle 和 raw 表中找到
PREROUTING	用来在包路由到其他链之前更改它们。可以在 nat、mangle 和 raw 表中找到
POSTROUTING	用来在包离开其他链之后即将走出接口时更改包。只能在 nat 和 mangle 包里找到

每个链与数据包通过主机时要走的基本路径相关联。当 Netfilter 遇到一个包时，它做的第一个评估是此包要去往哪个链。不是所有的表都包含表 6-5 所列的全部内置链。

让我们以 filter 表为例看一看。它只包含 INPUT、OUTPUT 和 FORWARD 链。如果一个包通过网络接口进入主机，那么它需要由 INPUT 链里的规则评估。如果此包由本机产生并且要通过网络接口出去，那么它需要由 OUTPUT 链里的规则评估。FORWARD 链用于已经进入主机但目的地是其他主机的包（例如在网络周边或你的网络与 Internet 之间担当路由器或者基于软件的防火墙的那些主机上）。在每个表中你可以创建自己的链来保存额外的规则。你还可以把包的流向从一个内置链导向自己创建的链，但是这只能在同一个表空间内进行。

提示：请把上述解释中的钩子看作一套鱼钩，这些鱼钩带各种类型的诱饵，以便钓不同类型的鱼。当附近游泳的鱼被某种诱饵所吸引，它就上钩了，那么它的命运就由鱼钩线末端的人决定。

6.4.3 策略

在过滤表中定义的每个内置链还可以有一个策略。策略就是一个链对数据包所采取的默认动作，它确定数据包在没有匹配链里任何规则时的操作。对数据包可以使用的策略是 DROP、REJECT 和 ACCEPT。当 iptables 命令第一次运行时，它为内置链设置一些默认策略。INPUT 和 OUTPUT 链的策略是 ACCEPT，而 FORWARD 链的策略是 DROP。

DROP 策略在不通知发送者的情况下丢弃包。REJECT 策略也丢弃包，但它给发送者发送一个 ICMP 包以告知丢弃事件。REJECT 策略意味着设备将知道它的包没有到达目的地，并迅速报告这个错误，而不是像使用 DROP 策略那样等待到超时。DROP 策略与 TCP RFC (Requests for Comment) 相反，可能对网络设备有点严厉；尤其是它们会长时间坐等那些被丢弃包的回应。但是为安全起见，一般认为使用 DROP 策略比 REJECT 策略要好，因为它给外部世界提供的信息更少。

ACCEPT 策略接受流量并允许它通过防火墙。自然地，从安全性的视角看，如果它被用作默认策略，那就表示防火墙无效了。默认情况下，iptables 为所有链配置了 ACCEPT 策略，但推荐把它改为 DROP 策略。

■注：在远程主机上把策略改为 DROP 之前，要确保已经配置了允许你连接的 ACCEPT 规则。

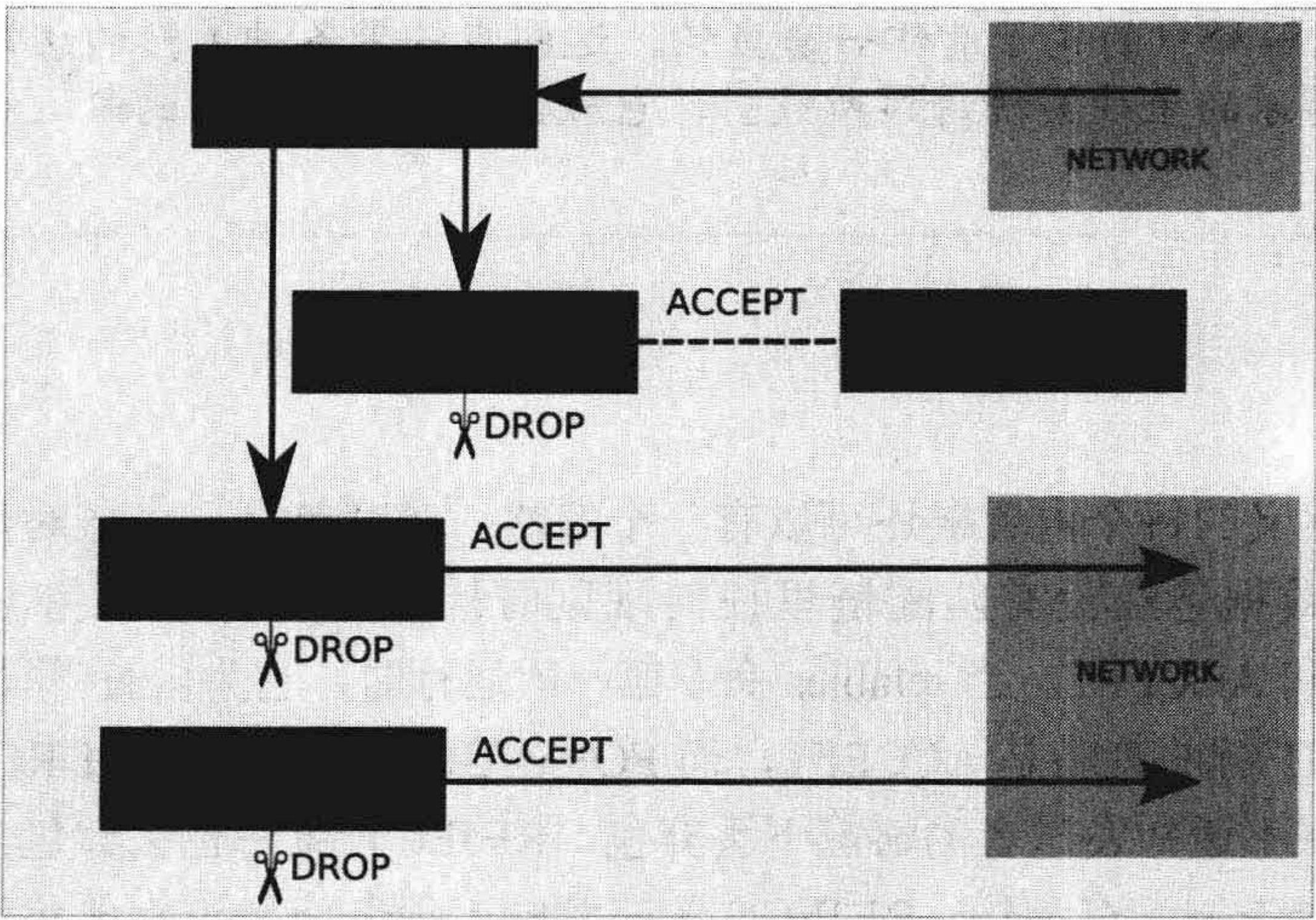
这就符合防火墙默认是拒绝姿态的基本原则。应该默认拒绝所有流量，只对明确授权的流量开放主机。这样拒绝可能有问题，因为把 INPUT 和 OUTPUT 链的默认策略设置为 DROP 意味着，如果没有明确添加允许流量进出主机的规则，进来的和出去的流量就都不被允许。这会导致所有从主机连接而没有明确允许进出该主机的服务和工具不能工作。

6.4.4 网络地址转换

网络地址转换让私有网络 IP 地址空间看起来源自单个公共 IP 地址。这可以利用 IP 伪装（重写 IP 包头）做到。NAT 把经过防火墙/路由器的连接信息保存在转换表里。一般来讲，当一个包从私有地址空间穿越服务器/路由器到公共地址空间时，转换表中就会写入一条输入项。当相应的包返回时，路由器就使用转换表把返回的包匹配给发起连接的源 IP。

许多年以前，Internet 上的每台主机都提供一个公共 IP 地址，以使它变成 Internet 的一分子并能与其他主机对话。人们很快就意识到，如果每个想连接 Internet 的主机都提供一个自有公共 IP 地址，那么可用的 IP 地址将被用完。NAT 的发明使私人地址空间可以看起来像来自单个 IP 地址。

对于一个单独的主机，通过主机防火墙的流量流程看起来如图 6-21 所示。



NETWORK：网络

图 6-21 通过单台主机防火墙的流量流向图

如果配置了 iptables，网络数据包就会通过防火墙接口进来。在图 6-21 中，流量从网络进入主机。网络可能是本地网络或者 Internet。每个包通过 Netfilter 程序时被直接传递，它的下一个路由由主机决定。做这个决定的根据是此包的最终目的地地址，看目的地是否是本地主机或者其他主机。

如果包目的地为其他主机而且本主机打开了 IP 包转发功能（在较早前的“添加路由与转

发数据包”一节解释过)，此包就会被发送给转发链。如果没有打开 IP 包转发功能，或者如果有拒绝通过主机转发的策略和规则，这个包就会被丢弃。一旦进入 FORWARD 链，如果有允许此包从主机转发的规则和策略，就允许它通过并发送到网络中。

如果包的最终目的地是防火墙自身，Netfilter 路由就把它传送给 INPUT 链。在这里，任何与所定义的规则不匹配的包都将根据默认策略处理。如果包与某个规则匹配，那么就可以用 -j target 选项的 target 指定一个动作。如果这个目标是 ACCEPT，那么此包继续在 IP 栈，允许此连接的建立和延续。

源自防火墙而目的地为其他主机的流量会传递给 OUTPUT 链。如果 OUTPUT 链的规则和策略允许这些包通过，它们就会从已配置接口出去。如果不允许，就根据默认的策略处理它们。

6.4.5 使用 iptables 命令

既然理解了数据包在 Netfilter 里的流程，那么现在就说明如何使用 iptables 命令管理它们。iptables 命令允许做下面的事情。

- 列出包过滤规则组的内容。
- 添加/移除/修改包过滤规则组里的规则。
- 列出包过滤规则组的总规则计数器，或者对该计数器清零。

为了有效地使用 iptables 命令，你必须拥有 root 的特权。基本的 iptables 命令结构和参数如下所示。

```
iptables -t table-name -command chain rulenumner paramaters -j target
```

表 6-6 给出了上述语法中的 command 字段（我们即将介绍它）的选项以及对它们的说明。

表 6-6 iptables 命令可用的 command 选项	
选项	说明
-L	列出内存中当前 iptables 的规则
-D	删除链里的一条规则。要删除的规则可以用规则号或者匹配的式样表示
-I	向链里插入一条规则。如果没有指定规则号，它会把规则插入链的顶端
-F	清空一个链。这会移除一个链里的所有规则或者一个表中所有链里的全部规则（如果没有指定链的话）
-A	往一个链追加一条规则。除了规则被默认添加到链的底端外，其他和 -I 一样
-X	删除一个链。链必须是空的并且在希望删除它时没有被其他链引用
-N	创建一个新链。不能与已有目标同名
-P	为一个链设置默认策略。每个内置链（INPUT、OUTPUT、FORWARD、POSTROUTING 等）都有一个默认的策略：ACCEPT、REJECT 或者 DROP

■注：在处理防火墙规则时，如果错误地阻止了所有网络流量，拥有对系统控制台的访问权总是一个好办法。如果犯了一个错误，通过网络特别是在远程系统上更改防火墙规则，可能会使人处于尴尬的处境。

iptables 命令的语法中的 parameters 有大量的参数可用来处理通过防火墙的数据包。我们解释链时说它们就像鱼钩。这些参数就是鱼钩上不同类型的鱼饵。每个参数都根据包中的信息吸引相应的包。它可能是包的源地址和目的地址，或者协议。包还可进一步用源端口、目的端口或者包的状态进行匹配。这些参数还可以用来匹配、捕捉、标记和释放信息包。

iptables 命令语法里-j target 所描述的 target 参数就是对钓上来的包所执行的动作。常见的 target 为 ACCEPT、DROP、LOG、MASQUERADE 和 CONNMARK。target 也可以是另一个用户自定义链。

现在要通过一些例子说明如何使用 iptables 命令。

iptables 命令是 Ubuntu 和 Red Hat 都有的命令，它们的操作方式一样。下面的例子说明如何管理 Red Hat 安装时所生成的 iptables 规则组。如果是 Ubuntu 主机，仍然建议阅读这部分内容，只是所显示的结果与 Ubuntu 上的结果不同而已。

让我们看一看当前正在运行的 iptables 规则组的内容。为此，我们发出下面的命令。

```
$ sudo /sbin/iptables -t filter -L --line-numbers
```

在这里查看的是 filter 表。-L 选项列出了所有的链以及与每个链相关的带行编号的规则(--line-number)。行编号很重要，因为就像前面提到的那样，iptables 按顺序从第一个到最后一个运行每个规则。当想添加一个新规则或者删除一个旧规则时，可以使用这些编号精确指明希望对哪一个规则操作。对于 Red Hat 主机，我们选择了一部分输出显示在列表 6-5 中。

列表 6-5 RH-Firewall-1-INPUT 链

```
Chain RH-Firewall-1-INPUT (2 references)
num target prot opt source destination
1 ACCEPT all -- anywhere anywhere
2 ACCEPT icmp -- anywhere anywhere icmp any
3 ACCEPT esp -- anywhere anywhere
4 ACCEPT ah -- anywhere anywhere
```

注意本例中显示了用户自定义的链 RH-Firewall-1-INPUT。Red Hat 主机在使用 system-config-security 工具或者作为安装的一部分配置防火墙时创建了这个链。

对于 Ubuntu 用户，iptables 会以没有任何 Ubuntu 预定义链的干净面目呈现。在 filter 表中，Ubuntu 在 INPUT 和 OUTPUT 链上的默认策略是 ACCEPT，在 FORWARD 链上的默认策略为 DROP。

```
$ sudo /sbin/iptables -L
Chain INPUT (policy ACCEPT)
target prot opt source destination

Chain FORWARD (policy DROP)
target prot opt source destination

Chain OUTPUT (policy ACCEPT)
target prot opt source destination
```

对最小规则需求的设置由你决定。不管怎样，通过发出下面的命令，你可以在 Ubuntu 主机上创建一个类 Red Hat 的相似链。


```
$ sudo /sbin/iptables -t filter -N Firewall-eth1-INPUT
```

该命令利用 `-N Firewall-eth1-INPUT` 在 `filter` 表中（由 `-t filter` 表示）创建一个新链。它还没有相关联的规则，不过到这一节结束你就会添加和删除规则了。继续使用 Red Hat 的规则组作为例子，但要知道同样的规则组可以应用到 Ubuntu 主机。

■注：不必使用前面所述的命名规则给链命名。不过，使用对配置有意义的名字会有好处。否则，`iptables` 规则输出看起来非常杂乱，将很难诊断故障。如果不愿意的话，也不需要创建自己的用户自定义链，但是如果创建的话，它会使规则更易读。

再回到列表 6-5，它显示了 `RH-Firewall-1-INPUT` 链，现在希望移除 `filter` 表中 `RH-Firewall-1-INPUT` 链的 `esp` 规则。

```
3 ACCEPT esp -- anywhere anywhere
```

为了做到这一点，要使用表名 `-t filter` 和要在其上操作的链 `RH-Firewall-1-INPUT`。使用 `-D` 选项是希望删除位置 3 的规则，如下所示。

```
$ sudo /sbin/iptables -t filter -D RH-Firewall-1-INPUT 3
```

现在再列出规则，这次把输出细化到刚好打印与 `RH-Firewall-1-INPUT` 链相关联的规则。

```
$ sudo /sbin/iptables -t filter -L RH-Firewall-1-INPUT --line-numbers
Chain RH-Firewall-1-INPUT (2 references)
num target prot opt source destination
1 ACCEPT all -- anywhere anywhere
2 ACCEPT icmp -- anywhere anywhere icmp any
3 ACCEPT ah -- anywhere anywhere
4 ACCEPT udp -- anywhere 224.0.0.251 udp dpt:mdns
```

现在可以看到与协议 `esp` 匹配的规则已经被移除，而与协议 `ah` 匹配的规则向上移到了编号 3 的位置。现在再把同样的规则添加进来，用 `-I` 命令把它插入到 `RH-Firewall-1-INPUT` 链编号 3 的位置，如 `-p esp` 选项所指示的那样，让它匹配 `esp` 协议，并给它提供 `-j ACCEPT` 的目标。

```
iptables -t filter -I RH-Firewall-1-INPUT 3 -p esp -j ACCEPT
```

现在再看 `RH-Firewall-1-INPUT` 链的第 3 行，这条规则已经插入并且目标为 `ACCEPT`。

```
Chain RH-Firewall-1-INPUT (2 references)
num target prot opt source destination
1 ACCEPT all -- anywhere anywhere
2 ACCEPT icmp -- anywhere anywhere icmp any
3 ACCEPT esp -- anywhere anywhere
4 ACCEPT ah -- anywhere anywhere
```

■注：封装安全载荷协议（Encapsulating Security Payload, ESP）和鉴别报头协议（Authentication Header, AH）都用于 IPsec 通信中。第 14 章将介绍更多有关它们的内容。

还可以清空链，意思就是移除所有链的全部规则或者指定链的全部规则。这可用于在添加一个新的规则组之前清除已有的规则。发出如下的命令可以达此目的。


```
$ sudo /sbin/iptables -t filter -F INPUT
```

这里清空了 (-F) filter 表中 INPUT 链的所有规则。如果现在查看 filter 表中的 INPUT 链，将看到已经没有任何与之关联的规则了。

```
$ sudo /sbin/iptables -L --line-numbers
Chain INPUT (policy ACCEPT)
num target prot opt source destination

Chain FORWARD (policy ACCEPT)
num target prot opt source destination
1 ACCEPT all -- anywhere 192.168.0.0/24 state RELATED,ESTABLISHED
```

在本例中，因为默认策略是 ACCEPT，所以清理这个链不会影响主机的工作方式。如果把此策略改成了 DROP，那么所有入站的连接都会被切断，从而造成主机上与网络相关的任务和服务不可用。这是因为此策略决定了不与任何规则匹配的包的命运。如果链里没有规则，那么 iptables 就使用默认策略处理包。

在主机上再创建下面的链。

```
$ sudo /sbin/iptables -t filter -N Firewall-eth1-INPUT
```

这里创建了链 Firewall-eth1-INPUT 而不是 Red Hat 默认的 RH-Firewall-1-INPUT。现在要给它添加一个规则。为了分析 Firewall-eth1-INPUT 链里的规则，需要在 INPUT 链里添加一个规则，让它把所有进入主机的包导向 Firewall-eth1-INPUT 链。

```
$ sudo /sbin/iptables -t filter -A INPUT -j Firewall-eth1-INPUT
```

这里发出的命令追加了这个规则。该规则表明，所有进入主机并以防火墙为最终目的地的包（如 -t filter -A INPUT 所示）都会被发送给 Firewall-eth1-INPUT 链（如 -j Firewall-eth1-INPUT 所示），以做进一步的处理。

通过添加其他参数可以细化希望在规则中抓住的东西。下面的例子只对 TCP 流量做匹配，并把它发送给某个目标。

```
$ sudo /sbin/iptables -t filter -A INPUT -p tcp -j Firewall-eth1-INPUT
```

可以进一步细化“诱饵”以吸引不同类型的鱼……呃……不同类型的包。这个命令在 INPUT 链上显示的结果揭示了下面的内容。

```
$ sudo iptables -L INPUT --line-numbers
Chain INPUT (policy ACCEPT)
num target prot opt source destination
1 Firewall-eth1-INPUT all -- anywhere anywhere
```

现在已经通过指定 -A 给 INPUT 链追加了一个规则。这个规则把来自所有源和目的地的全部流量都导向了用户自定义的 Firewall-eth1-INPUT 链。因为它是这个链里的唯一规则，所以它在规则组的位置编号是 1。

最后，在 INPUT 链上设置一个默认策略。我们把它设置为 DROP，以使所有与规则组不匹配的流量都自动“卧倒在地板上”。为达此目的，指定表名（用 -t filter 指定）、链（如 INPUT 所示）和目标（DROP）。

■注：这是会使所有网络连接掉线的潜在危险命令之一。在运行该命令之前，要确保自己可以访问实际的控制台，以防万一。或者可以使用像 <http://www.iptablesrocks.org/guide/safetynet.php> 里所描述的模式。

```
$ sudo /sbin/iptables -t filter -P INPUT DROP
```

如果现在列出 filter 表中的 INPUT 链，默认策略会显示为 DROP。

```
$ sudo iptables -L INPUT
Chain INPUT (policy DROP)
target prot opt source destination
Firewall-eth1-INPUT all -- anywhere anywhere
```

记得把 **DROP** 策略应用到所有链是一个好习惯。为了使防火墙最安全，就应该在所有链里强制默认策略为拒绝。

有了这些基本的命令，你就可以在自己的链和规则上执行大部分功能了。还有更多的功能可添加到规则中，从而使防火墙可以执行非常复杂而又有趣的路由任务。请查阅 iptables 的 man 页，那里有可执行任务的完整列表。

在 Red Hat 上开启和停止 iptables 很简单，在 Ubuntu 上稍微复杂些，除非使用像 ufw（本章稍后的“其他防火墙配置工具”一节会介绍它）那样的防火墙管理工具。

在 Red Hat 上，命令 service iptables 可以带几个参数。

```
$ sudo /sbin/service iptables start|stop|restart|condrestart|status|panic|save.
```

这里的参数 status、panic 和 save 不寻常。status 参数打印当前内存里的规则组。panic 参数会清空所有的规则并立刻把默认链上的所有策略设置为 DROP。它会阻止所有试图进出主机的连接，如果没有实际的应急需要，不要使用它。

save 参数告诉 iptables 把当前的规则组保存到/etc/sysconfig/iptables，以使对运行规则组的改动持久。

■注：如果使用 panic 选项，就会阻止到主机的所有流量。如果通过 SSH 或类似的工具远程连接到主机，连接也会掉线。如果没有任何其他访问主机的方法，你就会被关在外面，直到可以物理上登入主机修改防火墙规则或者让别人这么做。

在 Ubuntu 主机上，你可以使用 iptables-save 和 iptables-restore 命令找回 iptables 规则组并设置它。你可能希望把这些规则放进一个叫作/etc/networks/firewall 的文件里，然后从/etc/network/interfaces 文件调用它们。

下面是一个例子。

```
$ sudo /sbin/iptables-save > /etc/network/firewall
```

你可以对规则做任何改动和增删。然后你可以在/etc/network/interfaces 文件里使用下面的命令激活这些规则并重新保存它们。

```
auto eth0
iface eth0 inet dhcp
    pre-up iptables-restore < /etc/network/firewall
    post-down iptables-save -c > /etc/network/firewall
```


这里为 `eth0` 设置了标准的接口配置。在使接口为活动状态之前使用 `pre-up` 命令激活防火墙规则。然后使用 `post-down` 命令使接口不活动时保存规则。

如果你使用的是 Ubuntu 主机上的 `ufw`，那么可以使用 `invoke-rc.d` 命令开启和停止 `ufw`。

```
$ sudo invoke-rc.d ufw start|stop|restart
```

在这里 `invoke-rc.d` 用来管理 `ufw` 服务。本章稍后的“其他防火墙配置工具”一节会介绍 `ufw`。

进一步的阅读

`Netfilter` 和 `iptables` 是复杂的话题，推荐深入阅读相关内容。关于 `Netfilter` 和 `iptables` 命令的其他信息，可以阅读它的 `man` 页。如下所示的一些在线教程也可以利用。

- <http://iptables-tutorial.frozentux.net/iptables-tutorial.html>
- http://www.linuxtopia.org/Linux_Firewall_iptables/index.html

Ubuntu 还在本地硬盘上安装了一些 `iptables` 文档。

- `/usr/share/doc/iptables/html/packet-filtering-HOWTO.html`
- `/usr/share/doc/iptables/html/NAT-HOWTO.html`

至于更多高级话题，下面这些可能有用。

- *Designing and Implementing Linux Firewalls with QoS using netfilter, iproute2, NAT and L7-filter* by Lucian Gheorghe (PACKT Publishing, 2006)
- *Linux Firewalls: Attack Detection and Response with iptables, psad, and fwsnort* by Michael Rash (No Starch Press, 2007)

6.4.6 对 Red Hat 主机上默认规则的解释

正如前面所讲的那样，Ubuntu 在安装时不带任何默认的防火墙规则。而 Red Hat 不是这样。虽然不一定推荐 Red Hat 用它的 `system-config-security` 工具所创建的所有规则，但你可以使用这个工具为单机创建一个合理的防火墙，而且它提供了一个极好的有关 `iptables` 防火墙配置的概观。

■注：Ubuntu 读者仍然可以发现本节的趣味，因为我们讲述了 Red Hat 如何使用它的用户自定义链，而 Ubuntu 的 `iptables` 管理工具与 `ufw` 一样，也可以创建用户自定义链。

在 Red Hat 主机上可以使用 `service iptables status` 命令（这等同于在命令行上对每个表都使用 `iptables -t tablename -L --line-numbers -n` 命令）获得当前规则组的清单。列表 6-6 显示了 Red Hat 安装时没有附加规则的默认防火墙配置。

■注：如果 Ubuntu 用户使用一个类似 `ufw` 的 `iptables` 工具，他们会发现一组类似的文件。`/etc/ufw` 目录下的 `before.rules` 和 `after.rules` 文件包含规则组。

列表 6-6 Red Hat 默认的 iptables 规则组

```
$ sudo /sbin/service iptables status
Table: filter
Chain INPUT (policy ACCEPT)
```



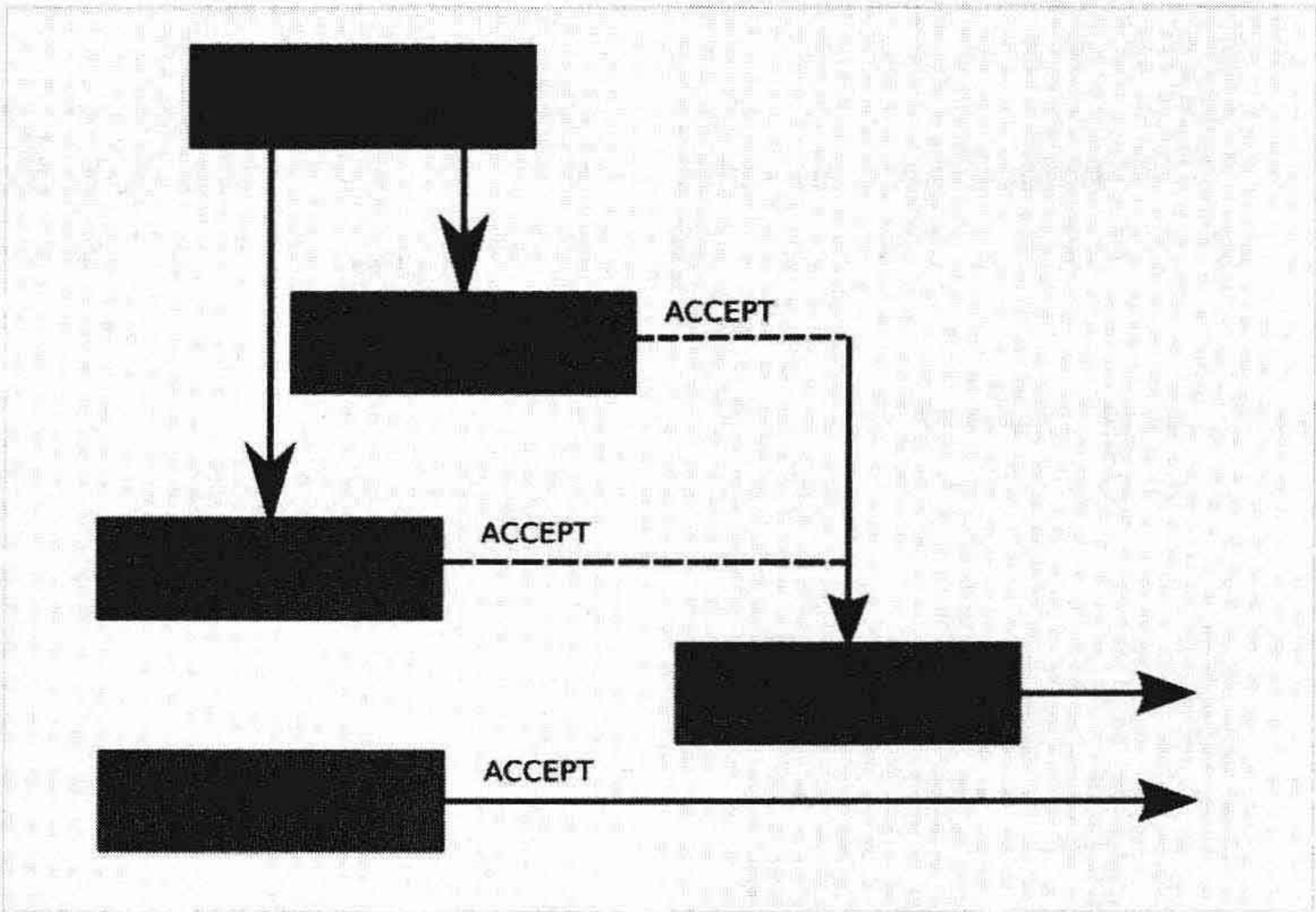
```
num target prot opt source destination
1 RH-Firewall-1-INPUT all -- 0.0.0.0/0 0.0.0.0/0

Chain FORWARD (policy ACCEPT)
num target prot opt source destination
1 RH-Firewall-1-INPUT all -- 0.0.0.0/0 0.0.0.0/0

Chain OUTPUT (policy ACCEPT)
num target prot opt source destination

Chain RH-Firewall-1-INPUT (2 references)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0
2 ACCEPT icmp -- 0.0.0.0/0 0.0.0.0/0 icmp type 255
3 ACCEPT esp -- 0.0.0.0/0 0.0.0.0/0
4 ACCEPT ah -- 0.0.0.0/0 0.0.0.0/0
5 ACCEPT udp -- 0.0.0.0/0 224.0.0.251 udp dpt:5353
6 ACCEPT udp -- 0.0.0.0/0 0.0.0.0/0 udp dpt:631
7 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:631
8 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
9 REJECT all -- 0.0.0.0/0 0.0.0.0/0 reject-with icmp-host-prohibited
```

从列表 6-6 中可以看到我们列了一个表 filter，它包含 3 个内置链：INPUT、FORWARD 和 OUTPUT。这些链都有 ACCEPT 的默认策略。你还可以看到另外一个链，叫作 RH-Firewall-1-INPUT，有两个链引用它（2 references）：INPUT 和 FORWARD。为了理解这里发生的事情，请看图 6-22，它显示了一个包是如何流过带上规则组的 filter 表的。



NETFILTER ROUTING: NETFILTER 路由
图 6-22 流量经过 filter 表的流程

从图 6-22 可以看到包流经 filter 表的方式。Netfilter 防火墙程序会根据进来的 IP 包的目的地把它们路由到 INPUT 或者 FORWARD 链。列表 6-6 中的 INPUT 规则说它会把所有包都传给 Firewall-1-INPUT 链，不管它的类型、源和目的是什么。如果包的目的地就是防火墙主机本身，它们会被传给 INPUT 链然后自动传给 RH-Firewall-1-INPUT 链。

如果进来的包目的地是其他主机，并且防火墙主机上的 IP 转发功能是打开的，它们就会被交给 FORWARD 链。在本例中，FORWARD 链只有一个规则，就是把所有包转发给 Firewall-1-INPUT。一旦通过了那个规则组的处理，这些包就会离开该主机（如果被允许）或者被阻止（如果不被允许）。

为什么把 INPUT 和 FORWARD 链指向 RH-Firewall-1-INPUT 链？Red Hat 希望同等对待 INPUT 和 FORWARD，因此把包导向同一个规则组。RH-Firewall-1-INPUT 本身的规则组很小，只允许一些非常基本的网络服务访问。我们可以在 INPUT 和 FORWARD 链上采用同样的规则，以达到同样的效果。

OUTPUT 链用于在当前的防火墙主机上发起目的地为其他地方主机的包。它的默认策略设置为 ACCEPT。这样就允许防火墙主机发出以其他任何主机为目的地的任何协议类型的套接字。

现在让我们看看希望在 Red Hat 主机上找到的 iptables 配置文件（在 Ubuntu 主机上，如果不创建，就没有 iptables 配置文件）。你可以直接从一个文件添加和编辑防火墙规则。你可以使用 iptables-save 命令把所有在内存里但还没有反映到/etc/sysconfig/iptables 文件的添加项包含进来。在这里，我们很快就会把当前的 iptables 规则保存到一个文件并查看它的内容。下面的命令会把当前的规则组保存到一个叫作 default-firewall_rules 的文件。

```
$ sudo /sbin/iptables-save > default-firewall_rules
```

在 Ubuntu 上，你可以使用相同的命令创建当前规则组的文件。

■注：不要直接编辑/etc/sysconfig/iptables 文件。这个文件本身有一个警告“不推荐人工定制本文件”。这是因为该文件是由 system-config-securitylevel 程序创建的，用户所做的任何改动会很容易就被它没有任何警告地覆盖掉。

列表 6-7 显示了文件 default-firewall_rules 的内容。可以看出这是安装 Red Hat 主机时所创建的一个文件。它已经有了几个输入项，我们将介绍它们。列表 6-7 添加了行号以有助于讲解该文件。

列表 6-7 default-firewall_rules 文件

```
1. # Generated by iptables-save v1.3.5 on Fri Jan 9 04:02:33 2009
2. *filter
3. :INPUT ACCEPT [0:0]
4. :FORWARD ACCEPT [0:0]
5. :OUTPUT ACCEPT [0:0]
6. :RH-Firewall-1-INPUT- [0:0]
7. -A INPUT -j RH-Firewall-1-INPUT
8. -A FORWARD -j RH-Firewall-1-INPUT
9. -A RH-Firewall-1-INPUT -i lo -j ACCEPT
10. -A RH-Firewall-1-INPUT -p icmp --icmp-type any -j ACCEPT
11. -A RH-Firewall-1-INPUT -p 50 -j ACCEPT
12. -A RH-Firewall-1-INPUT -p 51 -j ACCEPT
13. -A RH-Firewall-1-INPUT -p udp --dport 5353 -d 224.0.0.251 -j ACCEPT
14. -A RH-Firewall-1-INPUT -p udp -m udp --dport 631 -j ACCEPT
15. -A RH-Firewall-1-INPUT -p tcp -m tcp --dport 631 -j ACCEPT
16. -A RH-Firewall-1-INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
17. -A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited
18. COMMIT
```


在列表 6-7 的第一行里, #表示这是一行注释。接下来 *filter 表示正在处理的表: :INPUT、:FORWARD、:OUTPUT 和:RH-Firewall-1-INPUT 是链声明(第 3 行到第 6 行)。INPUT、FORWARD 和 OUTPUT 都有一个 ACCEPT 的默认策略。第 3 行到第 6 行的[0:0]是字节和计数器记号, 可用来查看多少流量和多少包通过了那个链(如果愿意的话, 你可以使用 iptables -Z chain 把这些计数器清零)。

第 7 行是 filter 表中 INPUT 链的第一个规则。如果想从命令行创建这一行, 我们可以发出下面的命令。

```
$ sudo /sbin/iptables -t filter -A INPUT -j RH-Firewall-1-INPUT
```

■注: 记住, 你可以使用 iptables 命令创建每一个规则, 或者把当前的规则组保存为一个文件并从日后恢复它的角度考虑编辑它。

前面已经介绍过它了。它做的事情是把通过 filter 表中 INPUT 链的所有包导向另一个叫作 RH-Firewall-1-INPUT 的链。

注意在第 8 行里, 所有通过 FORWARD 链的包也被导向 RH-Firewall-1-INPUT 链。

按照现在的情况, 这是一个相当不严格的防火墙, 因为默认策略为 ACCEPT。Red Hat 通过在 RH-Firewall-1-INPUT 链的结尾利用 REJECT 目标使之更加严格。现在看一看 RH-Firewall-1-INPUT 链里的规则。列表 6-7 里的第 9 行到第 17 行都是与此相关的规则。这些规则都是以从第一个到最后一个的次序分析的, 但是我们只想说明相关的核心规则。首先从最后的第 17 行开始。

```
-A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited
```

这是一个包罗甚广的规则, 应该总是放在这个链的最后一个位置。放在其后的任何规则都不会被分析, 因为之前没有匹配的包都被它拒绝了。因此当一个包经过此链时, 如果它不与任何规则匹配, iptables 就会用一个友好的管理性质的 ICMP 消息拒绝它。这等同于 INPUT 和 FORWARD 链上有一个 REJECT 的默认策略, 因为它们两者都把全部包传给 RH-Firewall-1-INPUT 链。然而, 正如前面所说的那样, 它和 INPUT 和 FORWARD 链上的 REJECT 默认策略绝对不同。

下一个重要的规则在第 16 行。这个规则使属于已经建立的连接或相关流量的连接得以接受。

```
-A RH-Firewall-1-INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

为了解释这一行, 让我们假设要发起一个到远程主机上的 DNS 服务器的出站连接。如前所述, iptables 对每个进出和经过防火墙的包的状态保持警觉。在发起 DNS 连接时, OUTPUT 链首先被分析, 以确保被允许建立与 DNS 服务器的连接。在防火墙的当前状态里有一个默认策略是 ACCEPT, 并且没有可分析的规则。我们可以发起到远程主机的套接字(NEW 状态)。远程 DNS 服务器会给套接字一个回应。所有入站的包都被发送给 filter 表的 INPUT 链。分析那个链里的规则, 然后传给 RH-Firewall-1-INPUT 链。在 RH-Firewall-1-INPUT 链里必须有一个规则接受这个输入的连接。如果在 RH-Firewall-1-INPUT 里有这样一个规则 -A RH-Firewall-1-INPUT -p tcp, udp -m state -state NEW --dport 53 -j ACCEPT, iptables 就会接受

它。一旦连接建立 (ESTABLISHED 状态), 防火墙就允许进行通信。防火墙记录连接的状态, 允许那些 ESTABLISHED 或者 RELATED 状态的连接继续。

补充知识: NEW、ESTABLISHED、RELATED 和 INVALID 状态

NEW 连接状态表示一个新发起的连接, 数据还没有通过它来回地传送。如果希望允许到某个服务的新连接, 就必须允许 NEW 连接状态。例如, 如果没有为某个邮件服务器上的输入 SMTP 流量指定接受 NEW 状态, 那么远程的客户端就不能使用该邮件服务器发送电子邮件。

ESTABLISHED 的连接状态表示一个正在传输数据过程中的已有连接。如果希望一个服务能维持和一个远程客户端或服务器的连接, 就要允许 ESTABLISHED 连接。例如, 如果想允许到主机的 SSH 连接, 就必须允许 NEW 和 ESTABLISHED 状态的输入流量和 ESTABLISHED 状态的输出流量, 以确保连接的可能性。

RELATED 状态是指用来为另一个连接提供方便的连接。一个常见的例子就是 FTP 会话, 在这种会话中, 控制数据通过一个连接传送, 而实际的文件数据通过另一个连接传输。

INVALID 状态标记在已经发现在处理包方面有问题的连接上: 它们可能已经超过了防火墙的处理能力或者它们是与当前任何连接都不相关的包。

■注: state 模块由 ipt_conntrack Netfilter 内核模块提供, 它在最近的大多数 iptables 发行版中都是默认加载。如果不是, 你可以使用 insmod 命令加载它: insmod ipt_conntrack。

第 9 行 -A RH-Firewall-1-INPUT -i lo -j ACCEPT 也很重要, 因为它接受源自回送地址 (lo, 或者 127.0.0.1) 的套接字连接, 以此来发起到它自身的连接 (因为 127.0.0.1 是不可路由的, 它只能连接到自身)。这很重要, 因为许多服务和应用程序使用回送地址发送信号或者连接到回送地址上的其他各种端口。

第 10 行 -A RH-Firewall-1-INPUT -p icmp --icmp-type any -j ACCEPT 允许进行 ICMP 连接, 包括如 ping、traceroute 以及其他的这类事情。这些连接用来传递路由信息或者所遇到的作为 RFC 792 (<http://tools.ietf.org/html/rfc792>) 和 RFC 1122 (<http://tools.ietf.org/html/rfc1122>) 一部分的错误报告。

第 11 行到第 15 行是 Red Hat 为了满足不同的常用网络需求而添加的。端口 50 和 51 被一个安全的加密传输隧道 IPsec 所使用。端口 5353 用于多播 DNS 服务, 这种服务允许本地网段中的主机回答特定的 DNS 查询 (更多的信息见 <http://www.multicastdns.org>)。最后, 端口 631 由 Internet 打印协议 (Internet Printing Protocol, IPP) 和 CUPS 打印服务使用。我们将从堡垒防火墙移除这些服务, 因为它们不是希望允许的那种服务 (不过第 14 章将重访 IPsec 连接)。对于堡垒主机, 关掉所有不想要的规则和服务特别重要。

```
-A RH-Firewall-1-INPUT -p 50 -j ACCEPT
-A RH-Firewall-1-INPUT -p 51 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 5353 -d 224.0.0.251 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 631 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 631 -j ACCEPT
```

最后, 第 18 行里的 COMMIT 是向内存确认那个表的规则组并激活规则组的指令。每个表都应该有一个 COMMIT; 否则它不会起作用。

6.5 配置范例网络

大部分网络都会有一些典型网络场景。它们在复杂性和安全需求上的程度不同。一个网络可以简单得只有两台用交叉线连接在一起的计算机，或者复杂得横跨全球，有集中的数据中心、灾难恢复站点和成千上万的服务器，每个服务器执行众多重要任务的一小部分。

本章余下的部分将示范如何配置早前所介绍的范例网络。本章开头说过，我们的网络由一个 Internet 连接、一个叫作 `gateway.example.com` 的 Linux 防火墙和一个叫作 `headoffice.example.com` 的主服务器组成。希望读者通过理解这个简单网络的构建方法，从中学到在任何主机上构建防火墙的方法，以及如何有一个模块化的安全计划，然后在需要时添加组件。

6.5.1 我们的配置

在这里将介绍本章开头简单描述过的范例网络组件。范例网络由以下组件组成。

- 1 个调制解调器/路由器。
- 1 个网络交换机/集线器。
- 1 个防火墙主机 `gateway.example.com`。
- 2 个以太网卡 `eth0` 和 `eth1`，用于 Linux 防火墙。`eth0` 接口上面是 PPP 通信，`eth1` 是内部私有网络接口。
- 1 个主要的商业服务器（组合了邮件、打印、文件和网站服务）`headoffice.example.com`。
- 1 组台式机/工作站。

在这个网络中，我们打算做几件稍微不同的事情。因为有了—个家庭或小型办公连接，所以 ISP 连接端接在调制解调器上，并为网络而非正要配置的专用防火墙处理 NAT 转换问题。对于使用有线调制解调器的家庭和小型办公网络，在我们的方案里，你应能够把 Linux 防火墙主机插入有线调制解调器。

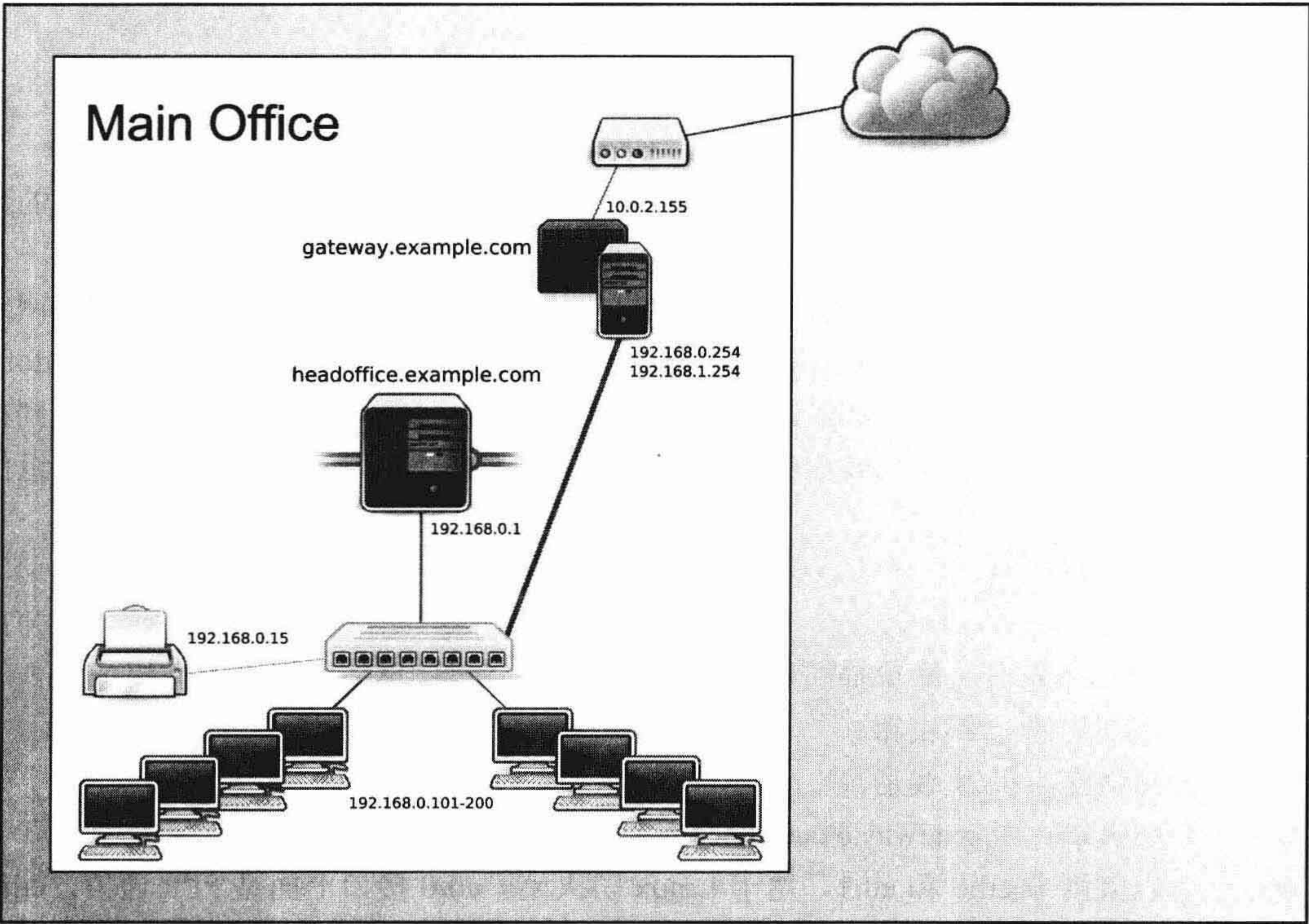
在一些情况下，如果调制解调器/路由器也包括一个防火墙，那么可以配置它，从而绕过—台 Linux 防火墙/路由器的需求，如果你这么想的话。不幸的是，由于可能使用的设备以及它们的配置方式之间存在很大的差别，我们不能讲述所有的操作方式。请查阅你的调制解调器厂商所提供的文档。

在我们的方案里，调制解调器/路由器处于桥接模式，因此来自 ISP 的 Internet 连接实际上端接在 Linux 防火墙主机上。看一看图 6-23，它再次显示了网络模样的示意图。

那么什么是桥接模式？桥接模式就是调制解调器并不试图与 ISP 建立点对点（Point-to-Point Protocol, PPP）的链路。它把来自 WAN（ISP 或者 Internet）的连接包传递给此连接的另一边或 LAN（在我们的例子中就是指一个运行着 PPPoE 后台程序的 Linux 主机）。然后这个后台程序可以与 ISP 网络建立一个连接。

如果调制解调器支持桥接模式，那么就要查阅厂商文档看如何操作。如果当前的调制解调器支持 IP 地址和 DNS，就必须首先把这些服务移到 Linux 主机上，因为调制解调器会丢

失它当前提供的 DNS 和 DHCP 服务。



Main Office: 主办公室
图 6-23 简单的网络示意图

■注：第 9 章将介绍 DNS 和 DHCP。

在桥接模式中，调制解调器只是毫不干涉地把连接转给 Linux 主机——它不提供防火墙功能，没有 NAT/SNAT（source network address translation，源网络地址转换），没有 DNS，也没有 DHCP。为了让 Internet 连接端接在 Linux 主机上，你需要添加一个 PPPoE 或者 PPPoA（Point-to-Point Protocol over Asynchronous Transfer Mode，异步传输模式的点到点协议）连接。然后 Linux 主机就变成网络中的路由器/防火墙。

■注：在这个网络范例中，使用 IPv4 私有地址范围内的 IP 地址 10.0.2.155 作为外部公共 IP 地址。在正常情况下，10.0.0.0/8 范围内的地址绝不会作为公共 IP 地址，因此不要在自己的网络中使用它。更多信息见 http://en.wikipedia.org/wiki/Private_network。

1. 建立一个 PPP 连接

在 gateway.example.com 堡垒主机或网关主机（它是网络周围路由器/防火墙的另一个名字）上，我们需要建立一个 PPP 服务。PPP 服务产生一个到 ISP 的连接，提供一些身份验证资料，并接收 ISP 所提供的 IP 地址。这个 IP 地址或者是一个静态地址，意思就是它不会变

动，或者是一个动态地址，意思就是它是从地址池中选取的，会每几天定期更换。我们将建立一个 PPP xDSL 连接，一种连接到 ISP 的普通方式。

在 Linux 上建立 xDSL 连接最普通的方式就是使用 Roaring Penguin PPPoE 客户端。为此，你首先需要验证已经安装了这个客户端。在 Red Hat 上，你需安装 `rp-pppoe` 软件包。在 Ubuntu 上，你必须安装 `pppoeconf`。

让我们简单地看一下如何着手用 Red Hat 和 Ubuntu 建立 PPP xDSL 连接。

在 Red Hat 上，有两种配置 PPP 连接的方式：在命令行下使用 `adsl-setup` 命令，或者使用 Red Hat 的 `system-config-network` GUI 程序。

在 Ubuntu 上，你可以像 Red Hat 上那样使用 `adsl-setup` 命令，在 `/etc/network/interfaces` 文件里创建一个 PPP 连接，或者使用 `network-admin` GUI。

使用 `adsl-setup` 设置 xDSL

既然 Red Hat 和 Ubuntu 都可使用 `adsl-setup` 命令，那么就介绍一下它，并通过一些基本的问答指导你设置连接。首先，它要求输入 ISP 所提供的 ISP 账号登录名。

```
$ sudo adsl-setup
Welcome to the ADSL client setup. First, I will run some checks on
your system to make sure the PPPoE client is installed properly...
```

LOGIN NAME

Enter your Login Name (default root): username@isp.com

接下来要求提供连接所在的接口。本例中的 PPP 连接需要一个以太网接口。调制解调器现在把来自 ISP 的 PPP 服务桥接到调制解调器上我们这边的设备。使用 `eth0` 接口（因为它被连接到调制解调器的集线器）作为本例的接口。

INTERFACE

```
Enter the Ethernet interface connected to the ADSL modem
For Solaris, this is likely to be something like /dev/hme0.
For Linux, it will be ethX, where 'X' is a number.
(default eth0): eth0
```

接下来，它问是否想要此连接按需求开启或者持续地开启。这是老的拨号调制解调器和 ISP 因按连接时间而不是按合理的下载量收费而存在的一个问题。我们要回答 `no`。

```
Do you want the link to come up on demand, or stay up continuously?
If you want it to come up on demand, enter the idle time in seconds
after which the link should be dropped. If you want the link to
stay up permanently, enter 'no' (two letters, lower-case.)
NOTE: Demand-activated links do not interact well with dynamic IP
addresses. You may have some problems with demand-activated links.
Enter the demand value (default no): no
```

下一个问题涉及 DNS：你希望提供域名服务器来解析 DNS 请求还是让 ISP 提供动态 DNS，又或者不希望这个配置涉及任何东西？在本例中，不输入任何信息，因为我们希望自己配置 `/etc/resolv.conf`（DNS 配置文件）。

DNS

Please enter the IP address of your ISP's primary DNS server.
If your ISP claims that 'the server will provide dynamic DNS addresses',
enter 'server' (all lower-case) here.
If you just press enter, I will assume you know what you are
doing and not modify your DNS setup.
Enter the DNS information here:

现在要输入 ISP 连接的密码。它是由 ISP 提供的。在屏幕上不会有任何回显信息，并且会要求再次输入该密码。

PASSWORD

Please enter your Password:
Please re-enter your Password:

接下来，它问是否愿意让普通用户控制连接的开启和停止。要说 no，意思就是必须使用 sudo 或 root 账号以 root 用户身份开启或停止连接。

USERCTRL

Please enter 'yes' (three letters, lower-case.) if you want to allow
normal user to start or stop DSL connection (default yes): no

现在，它问是否想在这个连接上设置一个防火墙。我们不打算使用它提供的防火墙设置，因为我们很快就会配置自己的防火墙，所以为接下来这个问题选择的答案是 0。

FIREWALLING

Please choose the firewall rules to use. Note that these rules are
very basic. You are strongly encouraged to use a more sophisticated
firewall setup; however, these will provide basic security. If you
are running any servers on your machine, you must choose 'NONE' and
set up firewalling yourself. Otherwise, the firewall rules will deny
access to all standard servers like Web, e-mail, ftp, etc. If you
are using SSH, the rules will block outgoing SSH connections which
allocate a privileged source port.

The firewall choices are:

- 0 - NONE: This script will not set any firewall rules. You are responsible
for ensuring the security of your machine. You are STRONGLY
recommended to use some kind of firewall rules.
- 1 - STANDALONE: Appropriate for a basic stand-alone Web-surfing workstation
- 2 - MASQUERADE: Appropriate for a machine acting as an Internet gateway
for a LAN

Choose a type of firewall (0-2): 0

最后，它问是否希望该服务在主机启动时开启。这会把配置文件中接口的 ONBOOT 选项设置为 yes。

Start this connection at boot time

Do you want to start this connection at boot time?
Please enter no or yes (default no):yes

现在呈现的是摘要信息，类似于下面的内容。如果这些设置符合你的意思，就输入 y。

```
** Summary of what you entered **
```

```
Ethernet Interface:  eth0
User name:           username@isp.com
Activate-on-demand:  No
DNS:                 Do not adjust
Firewalling:         NONE
User Control:        no
Accept these settings and adjust configuration files (y/n)? y
```

现在它已经在 `/etc/sysconfig/network-scripts/` 目录里创建了一个 `ifcfg-ppp0` 文件。前面讲过，`/etc/sysconfig/network-scripts` 是 Red Hat 主机存储网络配置文件的地方。`ifcfg-ppp0` 文件看起来类似以前所见的以太网文件，如列表 6-8 所示。

列表 6-8 Red Hat 上的 `ifcfg-ppp0` 文件

```
$ sudo cat /etc/sysconfig/network-scripts/ifcfg-ppp0
USERCTL=yes
BOOTPROTO=dialup
NAME=DSLppp0
DEVICE=ppp0
TYPE=xDSL
ONBOOT=yes
PIDFILE=/var/run/pppoe-adsl.pid
FIREWALL=NONE
PING=.
PPPOE_TIMEOUT=80
LCP_FAILURE=3
LCP_INTERVAL=20
CLAMPMSS=1412
CONNECT_POLL=6
CONNECT_TIMEOUT=60
DEFROUTE=yes
SYNCHRONOUS=no
ETH=eth0
PROVIDER=DSLppp0
USER=username@isp.com
PEERDNS=no
DEMAND=no
```

这应该不陌生；你可以看到该文件不同于以太网配置文件，但仍然有一些相同的变量和语法。我们把 `TYPE=xDSL`、设备名 `DEVICE=ppp0`、`BOOTPROTO=dialup` 和 `PROVIDER`、`USER` 选项设置为刚刚配置过的名字。有关所有这些变量的更多信息，请阅读 `man pppoe.conf` 页。

PPPoE 连接的密码现在存储在 `/etc/ppp/pap-secrets` 和 `/etc/ppp/chap-secrets` 中，如列表 6-9 所示。这个密码以明文的形式存储，要确保它受到适当的保护而不会被随意打探到。为此，我们需设置这些文件的安全权限。

```
$ sudo chown root:root /etc/ppp/pap-secrets /etc/ppp/chap-secrets
$ sudo chmod 0600 /etc/ppp/pap-secrets /etc/ppp/chap-secrets
```


列表 6-9 ppp0 的密码文件

```
$ sudo cat /etc/ppp/pap-secrets
# Secrets for authentication using PAP
# client      server secret          IP addresses
"username@isp.com" *      "myisp-password"
$ sudo cat /etc/ppp/chap-secrets
# Secrets for authentication using CHAP
# client      server secret          IP addresses
"username@isp.com" *      "myisp-password"
```

在 Ubuntu 上，一旦创建完一个 PPPoE 设备，你就会看到 ppp 部分已经被添加到了/etc/network/interfaces 文件。

```
iface ppp0 inet ppp
provider ppp0
auto ppp0
```

在这里你可以看到一个叫作 ppp0 的接口将在网络开启时创建（代码为 auto ppp0）。它是一个使用 PPP 的 IPv4 接口。接下来，注意设备信息是由 provider ppp0 提供的。提供者信息可在/etc/ppp/peers 目录下找到。Ubuntu 往/etc/ppp/peers/ppp0 文件的末尾追加了下面的信息。

```
plugin rp-PPPoE.so eth0
user username@isp.com
```

密码信息存储在/etc/ppp/chap.secrets 文件里，它的内容很像列表 6-9。

使用 GUI 设置 xDSL

我们很快就将介绍如何在 Red Hat 和 Ubuntu 上使用 GUI 工具配置一个 PPPoE 连接。

Red Hat

你同样可以使用 system-config-network GUI 配置 xDSL PPPoE 连接。与配置新以太网连接时要单击“新建”按钮一样，只不过这次选择的是 xDSL 连接，如图 6-24 所示。

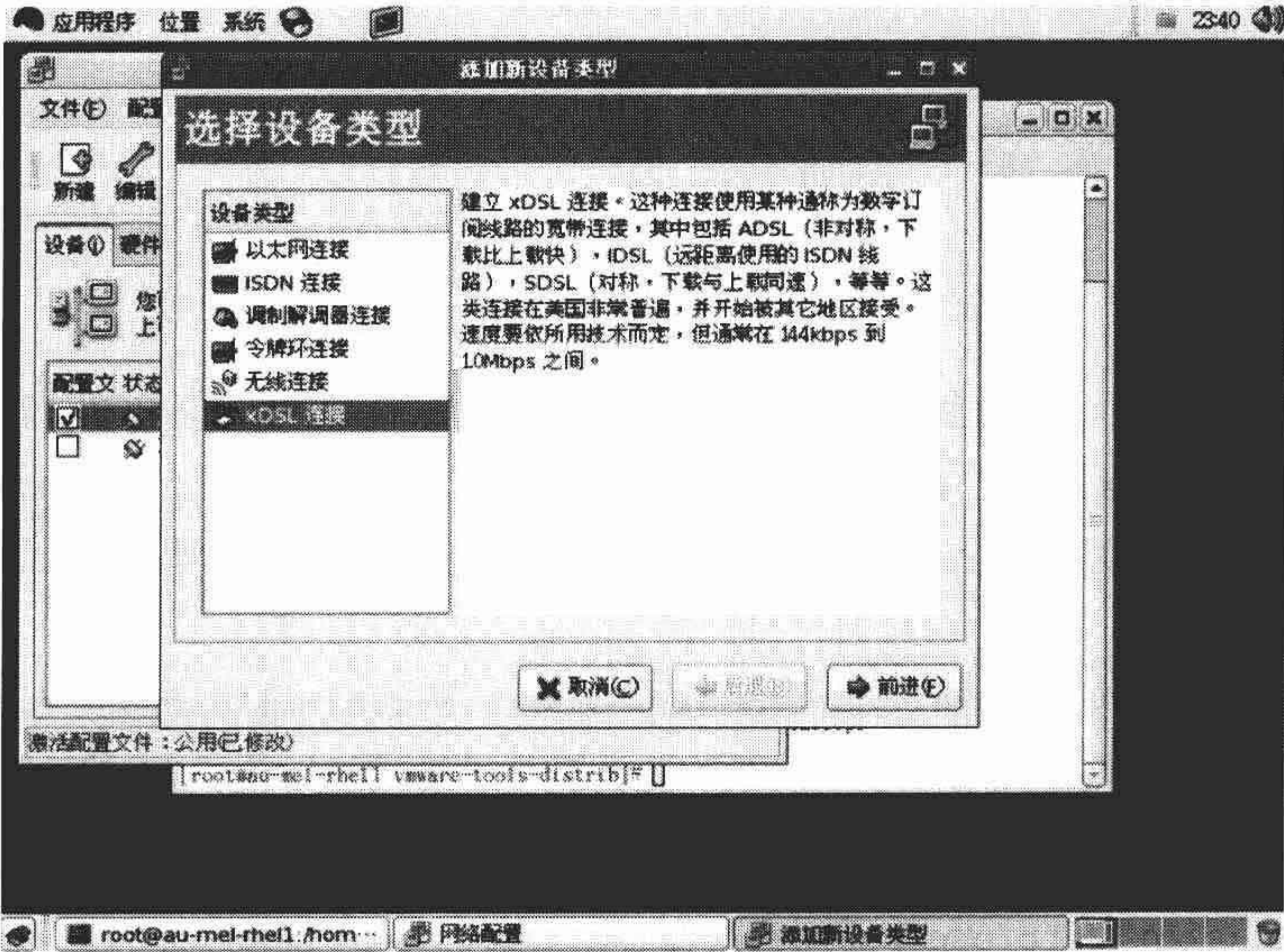


图 6-24 选择 xDSL 连接

接下来添加连接到 ISP 所需的资料。在提供者名字字段可以使用任何名字。你需添加 ISP 连接的用户名和密码，如图 6-25 所示。然后单击“前进”。



图 6-25 ISP 资料

现在呈现的是已输入资料的确认画面。单击“应用”就会把设备添加到配置中。你需保存此配置，然后才能激活它，这和设置普通以太网设备时要做的样。

如果想编辑连接资料，你就要再次突出显示此设备，本例中就是 ppp0，然后单击“Edit”图标。接下来你可以通过编辑刚创建的设备（见图 6-26）来更改任何资料。

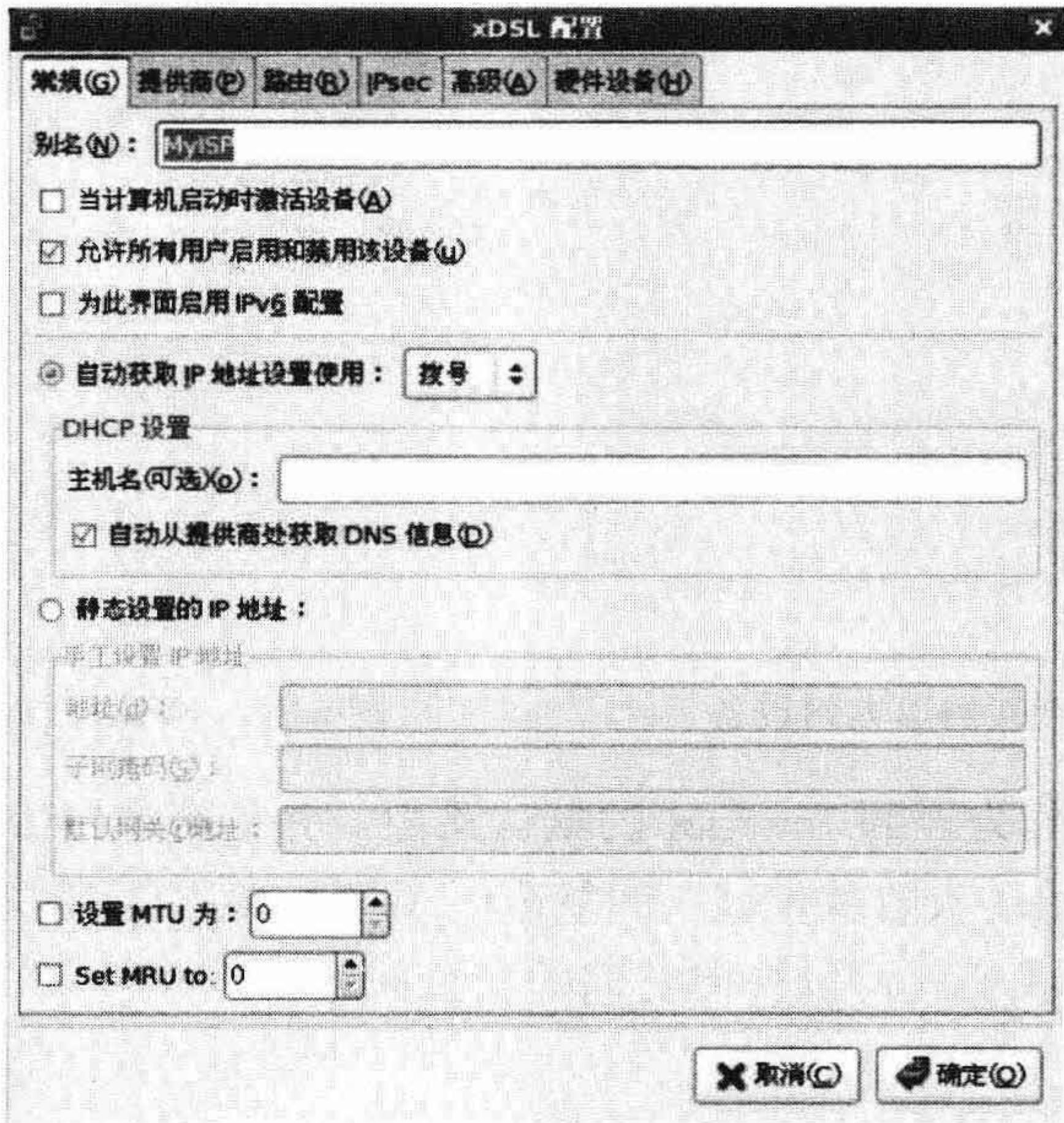


图 6-26 编辑资料

同样，你必须在改动发生前保存这些设置。主机现在已经为连接到 Internet 做好了准备。

它会在主机启动时尝试自动连接。

Ubuntu

在 Ubuntu 主机上，要用 Network Admin 工具创建设备。从图 6-27 可以看到一个待配置的 PPP 设备。单击“Properties”按钮可以编辑设置。

图 6-28 显示了出现的窗口。这里是添加 ISP 连接用户名和密码信息的地方。

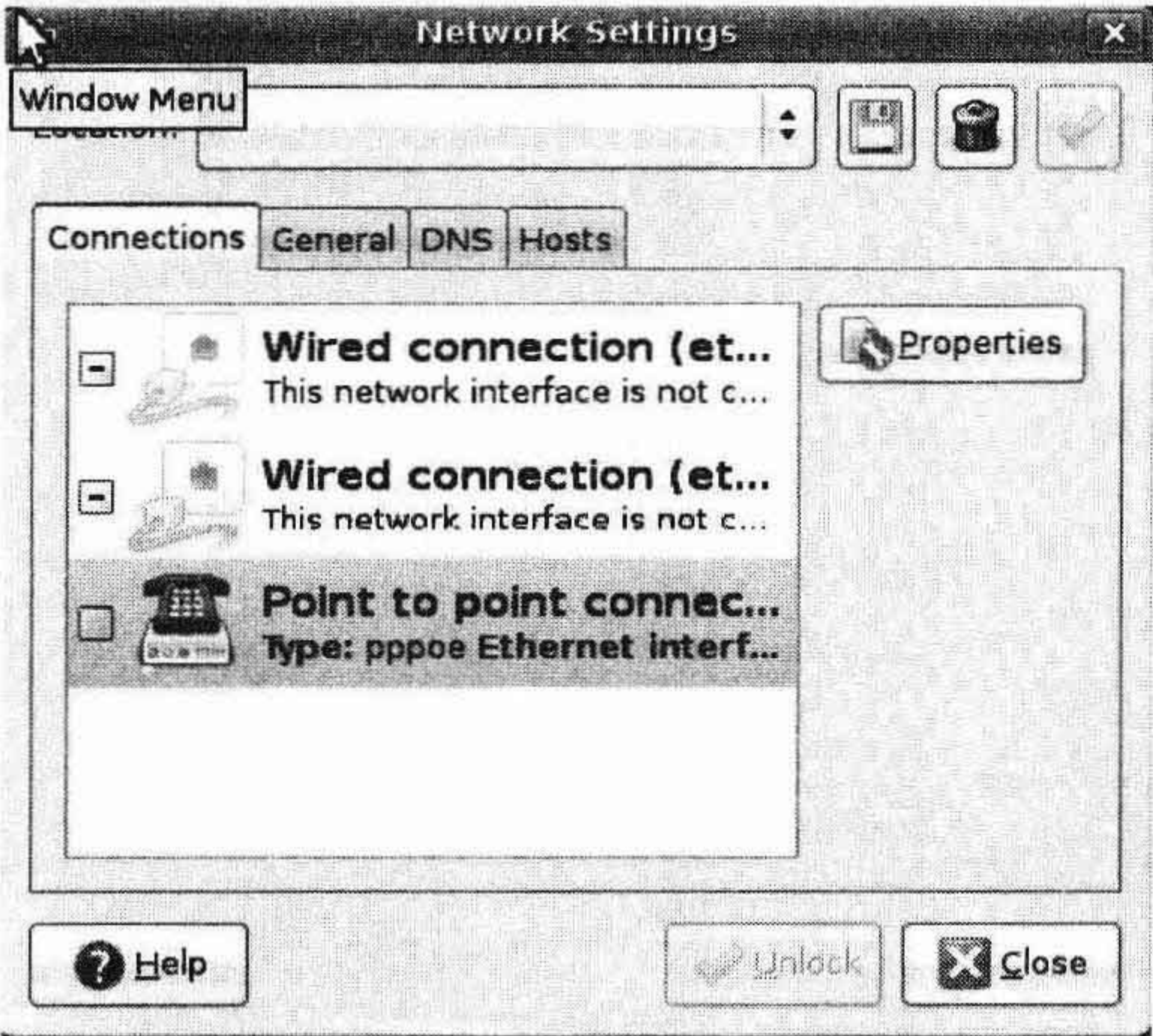


图 6-27 Network Admin 里的 PPP

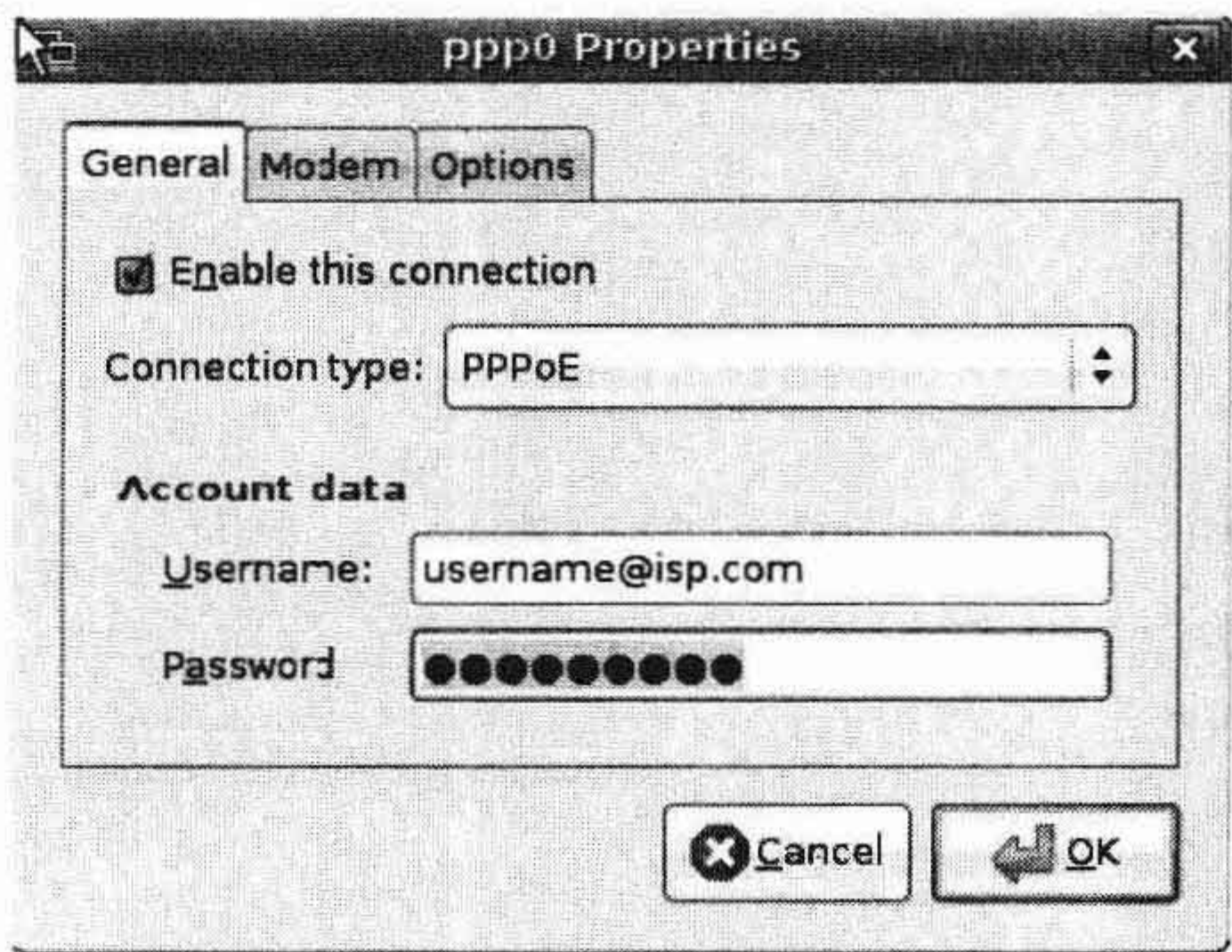


图 6-28 用户名和密码资料

现在选择希望这个 PPP 设备所依附的设备，如图 6-29 所示。
最后是一些有关连接的一般选项，如图 6-30 所示。

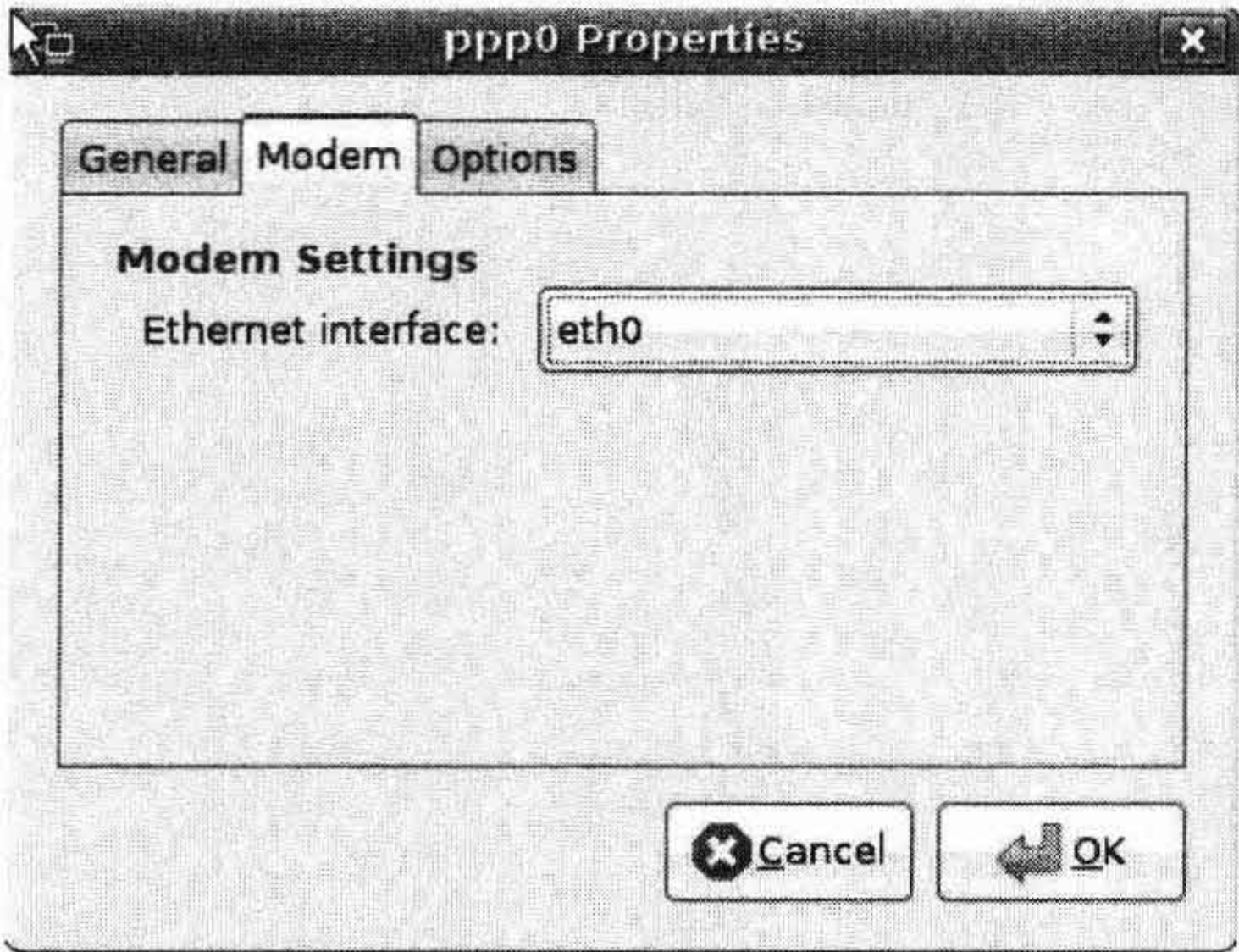


图 6-29 PPP 连接的以太网设备

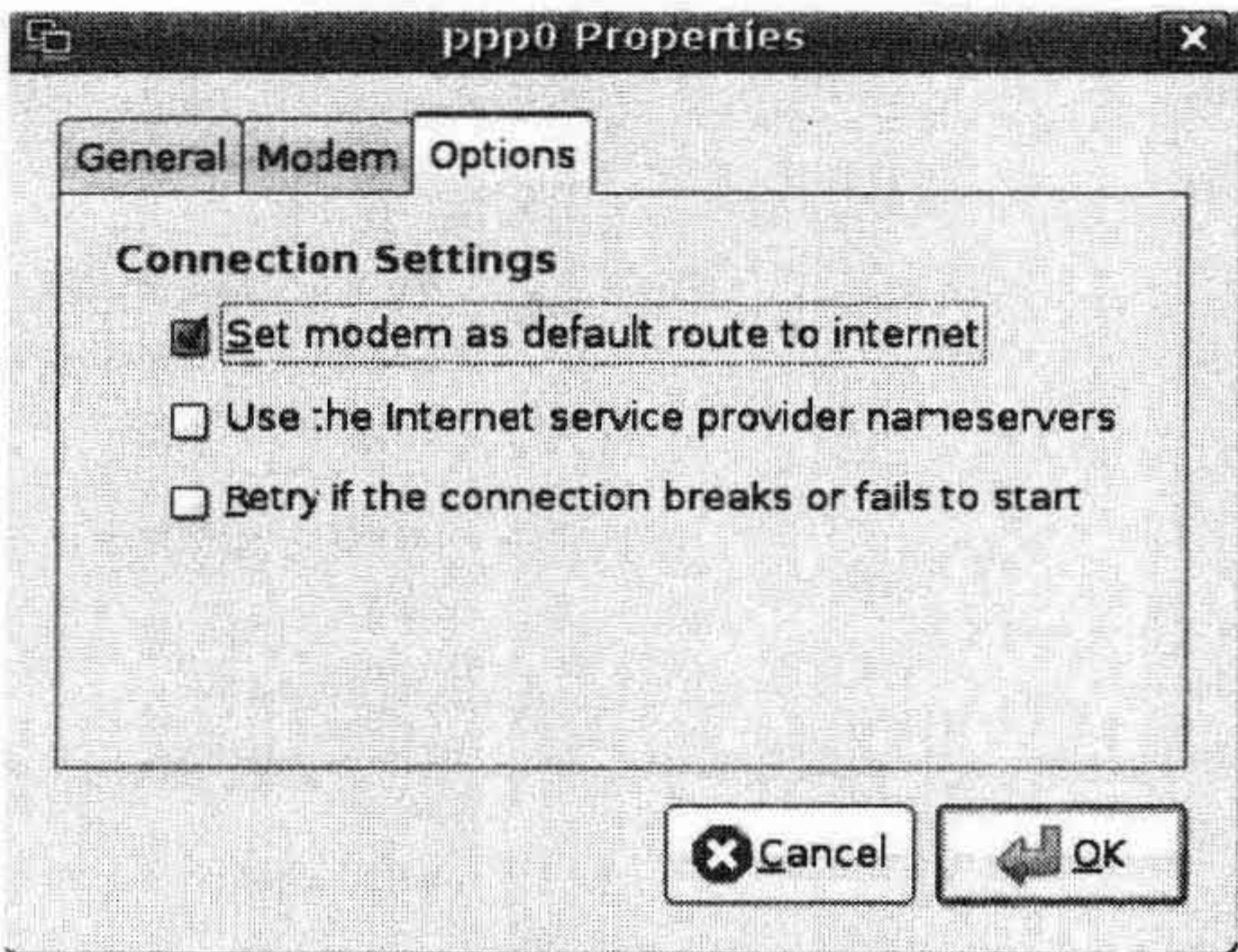


图 6-30 Network Admin 中的 PPP 选项

我们设置这个设备作为连到 Internet 的默认路由，并拒绝使用 ISP 的域名服务器，也没有选“如果连接中断或启动失败就重试”的复选框（图 6-30 中的第 3 个框）选项。既然已经设置了 PPP 连接，那么我们就可以转向它们的安全问题了。

2. 设置堡垒防火墙

上一节配置了 ppp0 接口，它处理到 ISP 的 PPP (oE) 连接。在配置期间，它问是否想在连接上配置一个防火墙，我们选择了不。现在就要配置防火墙。通过这样做，希望读者理解

如何配置一个防火墙，不仅对一个标准主机，还有网关主机。实际上，网关主机的不同在于它的路由功能，但是它仍需要为它自己的服务提供一个标准的防火墙配置。本节将逐步介绍它，先配置 filter 表，然后转到更多花样的防火墙功能。

为了能够配置一个功能全面又安全的防火墙，你要首先考虑主机将在网络中扮演什么角色，以及通过主机的典型期望流量。防火墙不能“即设即忘”。它们是动态的，要随着业务的改变而改变。不过，通过研究期望流量的流程，可以设想防火墙所需的一些基本的规则，以使它能够满足网络要求。

我们的防火墙主机有 2 个以太网卡来处理网络的不同部分。叫作 eth0 的设备附带有 PPPoE 设备 ppp0，它处理所有 Internet 或公共流量。叫作 eth1 的设备拥有私有网络或者 LAN 流量。如果防火墙上没有 2 个网卡，逻辑上通过在 iptables 配置里分隔 eth0 和 ppp0 设备仍然可以配置防火墙。

下面的两个表 6-7 和表 6-8 列出了流入主机和从主机流出的期望流量的流程。主服务器是唯一允许公共网络（Internet）流量进入公共主机（192.168.0.1）的主机。这是因为主服务器是网站服务器、邮件服务器和 DNS 服务器，所以需要来自 Internet 的访问。当为防火墙网关谋划网络流量时，要把期望连接映射到 192.168.0.1。现在搜集的这些数据也可以在为主服务器配置防火墙时使用。表 6-7 显示了什么流量会流出我们的网络。

表 6-7 流出网络的流量

接口	源 IP	协议	源端口	目的 IP	目的端口	用途
ppp0/eth1	192.168.0.1	TCP	Any	Any	25	SMTP
ppp0/eth1	192.168.0.1	TCP/UDP	Any	Any	53	DNS 或者域名
ppp0/eth1	192.168.0.1	TCP	Any	Any	123	时间或 NTP
ppp0/eth1	192.168.0.0/24	TCP	Any	Any	22	SSH
ppp0/eth1	192.168.0.0/24	TCP	Any	Any	80, 443	HTTP 或 www

表 6-7 提供了流出网络的期望流量、目的地资料和流量的用途清单。我们知道主机 192.168.0.1 是 1 个 SMTP、1 个 DNS、1 个 NTP 和 1 个 HTTP 服务器，并且那些流量会流经防火墙。私有网络上的所有主机都提供了在端口 22、80 和 443 上向任何公共主机发起连接的能力，因为即时业务需要这么做。稍后在探索更多业务所需的端口时会扩展这个列表。

■注：所提到的这些端口就像收音机频道频率。它们是预先定义的电台，在那里主机可以收听到某种流量。有网站流量的“电台”，也有发邮件的“电台”。与广播电台一样，这些端口不是随机的，而是由某个外部团体分配的。所分配的端口列表可以在/etc/services 文件里或者管理分配这些端口的组织（<http://www.iana.org/assignments/port-numbers>）找得到。

如果主机不在适当的位置，就很难准确地确认“线上”（进入接口的实际流量的另外一种提法，在本例中此接口是 eth0）发生了什么。这里所做的假设纯粹基于期望网络所发生的事情，而不是它实际上会发生的事情。一旦防火墙主机就位，我们就可以通过一个包监听工具（比如早前讲过的 tcpdump 或 Wireshark）分析流量，以进一步澄清和修补流量流程表，更好地配合我们的网络。然后我们可以使用这个信息进一步设置 Netfilter 规则，记录和阻止在这

个过程中注意到的那些不必要的流量。

表 6-8 显示了与表 6-7 相反的流量流程，这是我们期望看到进入我们网络的流量。它允许公共域的主机发起到私有网络的套接字。这里没有看到一个长列表，因为我们不是在提供许多对公服务。我们允许公共主机对 DNS、SMTP 和 HTTP 服务发起套接字。同样，这个列表可根据业务需要增长。

表 6-8 进入我们网络的流量流程

接口	源 IP	协议	源端口	目的 IP	目的端口	用途
ppp0	Any	TCP	Any	192.168.0.1	25	SMTP
ppp0	Any	TCP/UDP	Any	192.168.0.1	53	DNS 或域名
ppp0	Any	TCP	Any	192.168.0.1	80, 443	HTTP 或 www
eth1	192.168.0.0/24	TCP	Any	192.168.0.254	22	SSH
eth1	192.168.0.0/24	TCP/UDP	Any	192.168.0.254	53	DNS 或域名

从表 6-8 来看，同样明显的一件事就是需要实现目的网络地址转换（destination network address translation, DNAT），以使到公共接口地址上的端口连接映射到内部主机上的端口（常称作端口映射）。这是较流行的防火墙功能，本章最后会涉及它。

公共网络上的主机在尝试与，比如说，网站服务器建立一个套接字时，会尝试和外部或公共地址（10.0.2.155）上的端口 80 建立一个套接字，因为那是 DNS 提供给它们的地址。它们对内部网络一无所知，因此不知道实际上在 192.168.0.1 主机上提供网页服务，它们也不应该知道这些。防火墙主机本身不会回答它们的请求，因为它根本不会运行网络服务器。那么为了传播这个信息，我们使用防火墙把那些连接映射（或路由）到 192.168.0.1 主机。因此，由一些公共主机发起的到 10.0.2.155:80 的套接字会在内部端口映射到 192.168.0.1:80，就好像它实际上在和 10.0.2.155 对话。

■注：如果愿意的话，你可以在非标准端口上运行服务。在上述例子里，如果把 www 站点存储在 192.168.0.1:8080 上，那么还要把 10.0.2.155:80 映射到 192.168.0.1:8080。在这种情况下，所有在端口 80 发起的套接字都会在内部主机上被映射到端口 8080。

既然我们已经收集了期望网络流量如何流动的信息，现在就可以开始定义规则了。本例使用 Red Hat 主机作为堡垒防火墙。如前所述，这里给出的信息可以同等地应用于 Ubuntu 主机。

补充知识：链的命名

本章要创建用户自定义链和规则组，所采用的方式与 Red Hat 主机类似。例如，Red Hat 把它的 INPUT 和 FORWARD 链定义为 RH-Firewall-1-INPUT。在我们的改动中，会使用命名规则 Firewall-nic-INPUT，其中的 nic 是正要为之设计规则的网络接口卡（network interface card, NIC），比如 eth0、eth1 或 ppp0。我们希望用这种方式使规则组清楚它们要影响的流量。

当安装 Red Hat 主机并把防火墙的设置作为安装过程的一部分时，一组 iptables 规则就生成了。用户自定义链从根本上来说是一个好主意，并且对于单机来说足够了。因为它们产生的规则也许不是我们想要的，所以用户定义自己的用户链的概念很好。然而，这些链在有多个 NIC 时也会变得有限。

如果有一个名为 Firewall-ppp0-FORWARD 的链和它控制之下与那些流量相关的规则，那么我们很快就可以看到设备 ppp0 上 FORWARD 链里的流量。我们移除了这个链的 RH 前缀以及其他内容，以使它们不要太 Red Hat 中心化。实际要用的命名惯例完全由用户决定。不过，如果用类似的方式定义规则组，会让别人更易读。

为什么不为用户自定义链选择 Ubuntu 的命名惯例呢？因为 Ubuntu 在安装过程中不创建规则组，它没有任何规则和用户自定义链。如 ufw 这样的工具也可以创建用户自定义链，如果更喜欢它的话，你可以自由遵循那个命名规则。只要规则清楚易读，选择哪个命名惯例都没关系。

我们开始先把当前规则组保存为一个文件。我们在命令行里发出下面的命令。

```
$ sudo /sbin/iptables-save > firewall-20090101
```

这将把当前内存里的 iptables 规则保存到文件 firewall-20090101。

通过发出下面的命令我们可以在某个稍后的日期恢复保存过的规则组。

```
iptables-restore < firewall-20090101
```

它会从文件 firewall-20090101 恢复防火墙规则并把新的防火墙规则保存到/etc/sysconfig/iptables。

创建自己的规则：基本的 filter 表

我们现在想开始编辑、创建自己的防火墙规则，因为已经研究了网络需求，已经看到了防火墙规则的当前状态。首先，我们把当前规则的一个副本保存在一个文件中，去除不合需要的规则，并开始写自己的规则组。

我们需要创建一个文件来保存规则组。我们给这个文件起一个有意义的名字，如下所示。

```
$ sudo vi /home/jsmith/gateway.example.com-20090101
```

现在我们可以编辑文件/home/jsmith/gateway.example.com-20090101。我们从添加下面的规则组开始。

```
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
:Firewall-eth1-INPUT - [0:0]
-A INPUT -j Firewall-eth1-INPUT
-A FORWARD -j Firewall-eth1-INPUT
-A Firewall-eth1-INPUT -i lo -j ACCEPT
-A Firewall-eth1-INPUT -p icmp --icmp-type any -j ACCEPT
-A Firewall-eth1-INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A Firewall-eth1-INPUT -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

所有的链上都设置了一个 DROP 默认策略。我们希望所有的链默认采取拒绝（按照例外许可原则）。我们希望确保如果没有在防火墙规则里明确指定，任何事情都不被允许。

现在有一个新的 filter 表，所有内置链的策略都设置为 DROP。在当前配置下，主机不能与其他任何主机通信。那么让我们从设置防火墙主机规则开始。

现在有几件事要考虑，如两个接口设备 `eth1` 和 `ppp0`。`eth1` 设备是一个私有网络接口，`ppp0` 是公共接口。我们要把流量分成两个范围：一个处理内部私有流量，一个处理外部公共的流量。

我们正在关注进入主机的流量。我们确定要在防火墙上为私有网络打开下面的端口：在私有网络的端口 22 上通信的 SSH，以及在私有和公共两个网络的 TCP 和 UDP 端口 53 上通信的域名服务。我们需要回送地址以便能够给自己发起套接字，并且允许 ICMP 包的答复。相关或建立状态的所有包也都被接受。

```
-A INPUT -i eth1 -j Firewall-eth1-INPUT
-A FORWARD -i eth1 -o eth1 -j Firewall-eth1-INPUT
-A Firewall-eth1-INPUT -i lo -j ACCEPT
-A Firewall-eth1-INPUT -p icmp -m icmp --icmp-type any ➡
-m limit --limit 3/s -j ACCEPT
-A Firewall-eth1-INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A Firewall-eth1-INPUT -s 192.168.0.0/24 -p tcp ➡
-m state --state NEW -m tcp --dport 22 -j ACCEPT
-A Firewall-eth1-INPUT -p tcp -m state --state NEW -m tcp --dport 53 -j ACCEPT
-A Firewall-eth1-INPUT -p udp -m state --state NEW -m udp --dport 53 -j ACCEPT
-A Firewall-eth1-INPUT -j REJECT --reject-with icmp-host-prohibited
```

现在已做了什么事情？呃，在第一行，我们把所有从 `eth1` 进来的流量导向 `Firewall-eth1-INPUT` 链。

```
-A INPUT -i eth1 -j Firewall-eth1-INPUT
```

接下来，我们把通过主机转发的包，即那些从接口 `eth1` 进入且通过 `eth1` 退出的包，也导向 `Firewall-eth1-INPUT` 链。

```
-A FORWARD -i eth1 -o eth1 -j Firewall-eth1-INPUT
```

然后我们允许回送地址发起连接，允许 ICMP 包，并且通过 `-m limit -limit 3/s` 限制接受的数量只能是每秒 3 个。本章稍后的“配置我们的规则：重访 filter 表”一节将深入解释这一点。

```
-A Firewall-eth1-INPUT -i lo -j ACCEPT
-A Firewall-eth1-INPUT -p icmp -m icmp --icmp-type any ➡
-m limit --limit 3/s -j ACCEPT
```

我们还希望允许接受路过的 `RELATED` 和 `ESTABLISHED` 状态的包。

```
-A Firewall-eth1-INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
```

下面一行允许从私有网络上的源地址到防火墙主机上的 SSH 连接，即 `-s 192.168.0.0/24`。如果我们指定 `192.168.0.*/32`（其中 `*` 是 IP 地址的最后一个 8 位字节），就可以把它限制为单台主机。

```
-A Firewall-eth1-INPUT -s 192.168.0.0/24 -p tcp -m state ➡
--state NEW -m tcp --dport 22 -j ACCEPT
```

我们还打开了到防火墙主机的 DNS 连接，即 `--dport 53`，并期待任何主机都可以从私有或公共网络查询 DNS 服务器，因此没有指定目的地网络的源。DNS 实际上通过 TCP 和 UDP 两个协议提供服务，因此对每个协议都在单独的一行指定。这些协议可以在单行里指定，用逗号分开，就像这样：`-p tcp, udp`。


```
-A Firewall-eth1-INPUT -p tcp -m state --state NEW -m tcp --dport 53 -j ACCEPT
-A Firewall-eth1-INPUT -p udp -m state --state NEW -m udp --dport 53 -j ACCEPT
```

我们还希望拒绝与这几个规则不匹配的包，并发出一个 ICMP 消息告诉正尝试连接的服务。

```
-A Firewall-eth1-INPUT -j REJECT --reject-with icmp-host-prohibited
```

我们把所有内置链的默认策略设置为 **DROP**，包括 **OUTPUT** 链。我们希望确保防火墙可以与私有网络通信，那么现在就添加下面的规则。

```
-A OUTPUT -s 192.168.0.0/255.255.255.0 -d 192.168.0.0/255.255.255.0 -j ACCEPT
```

我们在这个规则中使用了源参数（即 `-s 192.168.0.0/255.255.255.0`）和目的参数（在这里就是 `-d 192.168.0.0/255.255.255.0`），从而限定可以发起连接的主机。我们当然可以使用 `-i eth1` 和 `-o eth1` 代替指定 IP 网络，这是较宽松的约束。

稍后的“配置我们的规则：重访 filter 表”一节将进一步约束 **OUTPUT** 链，让它只允许由防火墙发起所期望的通信。

让我们在这里停一会。如果回顾一下当前文件中的规则，我们就会看到已经有了主机防火墙框架。所有内置链上的默认策略都是 **DROP**，并且有一组只允许所期望的通信类型的规则。下面的规则可以复制、粘贴下来，我们将来可以使用它们作为主机上的防火墙基础。

```
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
-N Firewall-eth1-INPUT
-A INPUT -i eth1 -j Firewall-eth1-INPUT
-A FORWARD -i eth1 -o eth1 -j Firewall-eth1-INPUT

-A Firewall-eth1-INPUT -i lo -j ACCEPT
-A Firewall-eth1-INPUT -p icmp -m icmp --icmp-type any ➡
-m limit --limit 3/s -j ACCEPT
-A Firewall-eth1-INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A Firewall-eth1-INPUT -p tcp -m state --state NEW -m tcp --dport 53 -j ACCEPT
-A Firewall-eth1-INPUT -p udp -m state --state NEW -m udp --dport 53 -j ACCEPT
-A Firewall-eth1-INPUT -s 192.168.0.0/24 -p tcp ➡
-m state --state NEW -m tcp --dport 22 -j ACCEPT
-A Firewall-eth1-INPUT -j REJECT --reject-with icmp-host-prohibited

-A OUTPUT -s 192.168.0.0/255.255.255.0 -d 192.168.0.0/255.255.255.0 -j ACCEPT
COMMIT
```

为了创建 `headoffice.example.com` 主机所需的防火墙，我们需要做的全部事情就是添加这些规则以允许 DNS、DHCPD、NTP、SMTP、IMAP 和 HTTP 服务连网。第 9 章、第 10 章和第 14 章将详述这些服务。

headoffice.example.com 防火墙

主机 `headoffice.example.com` 是主服务器，有许多用于内部办公和公共客户端的服务运行在上面。我们需要邮件服务（SMTP 和 IMAP）和网络服务（HTTP 和 HTTPS）以供公共网络和私有网络的访问，并限定 DNS、DHCPD 和 NTP 只为内部私有网络服务。

我们往基本规则组里添加下面的内容。

```
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
-N Firewall-eth1-INPUT
-A INPUT -i eth1 -j Firewall-eth1-INPUT
-A FORWARD -i eth1 -o eth1 -j Firewall-eth1-INPUT

-A Firewall-eth1-INPUT -i lo -j ACCEPT
-A Firewall-eth1-INPUT -p icmp -m icmp --icmp-type any ➡
-m limit --limit 3/s -j ACCEPT
-A Firewall-eth1-INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A Firewall-eth1-INPUT -p tcp -m state --state NEW --dport 53 -j ACCEPT
-A Firewall-eth1-INPUT -p udp -m state --state NEW --dport 53 -j ACCEPT
-A Firewall-eth1-INPUT -s 192.168.0.0/24 -p tcp -m state ➡
--state NEW --dport 22 -j ACCEPT
-A Firewall-eth1-INPUT -p tcp -m state --state NEW --dport 25 -j ACCEPT
-A Firewall-eth1-INPUT -s 192.168.0.0/16 -p udp -m state ➡
--state NEW --sport 68 --dport 67 -j ACCEPT
-A Firewall-eth1-INPUT -s 192.168.0.0/16 -p tcp -m state ➡
--state NEW --sport 68 --dport 67 -j ACCEPT
-A Firewall-eth1-INPUT -p tcp -m state --state NEW --dport 80 -j ACCEPT
-A Firewall-eth1-INPUT -s 192.168.0.0/24 -p udp -m state ➡
--state NEW --sport 123 --dport 123 -j ACCEPT
-A Firewall-eth1-INPUT -s 192.168.0.0/24 -p tcp -m state ➡
--state NEW --dport 143 -j ACCEPT
-A Firewall-eth1-INPUT -p tcp -m state --state NEW --dport 443 -j ACCEPT
-A Firewall-eth1-INPUT -j REJECT --reject-with icmp-host-prohibited

-A OUTPUT -s 192.168.0.0/24 -d 192.168.0.0/24 -j ACCEPT
COMMIT
```

这里添加的规则允许访问 **headoffice.example.com** 上运行的服务。DNS 被允许通过下面的规则访问，它接受来自私有网络和公共网络的连接。

```
-A Firewall-eth1-INPUT -p tcp -m state --state NEW --dport 53 -j ACCEPT
-A Firewall-eth1-INPUT -p udp -m state --state NEW --dport 53 -j ACCEPT
```

这个规则提供了从本地网络内部访问 SSH 的能力。

```
-A Firewall-eth1-INPUT -s 192.168.0.0/24 -p tcp -m state ➡
--state NEW --dport 22 -j ACCEPT
```

我们可以从拥有下面这个规则的任何主机上进行邮件访问——SMTP。

```
-A Firewall-eth1-INPUT -p tcp -m state --state NEW --dport 25 -j ACCEPT
```

DHCP 需要下面的规则来打开对本地网络的访问。注意我们特别允许对所有子网掩码为 192.168.0.0/16 的子网访问。

```
-A Firewall-eth1-INPUT -s 192.168.0.0/16 -p udp -m state ➡
--state NEW --sport 68 --dport 67 -j ACCEPT
-A Firewall-eth1-INPUT -s 192.168.0.0/16 -p tcp -m state ➡
--state NEW --sport 68 --dport 67 -j ACCEPT
```


对 HTTP 和 HTTPS 的访问通过下面两个规则授权。我们允许来自所有网络的访问。

```
-A Firewall-eth1-INPUT -p tcp -m state --state NEW --dport 80 -j ACCEPT
-A Firewall-eth1-INPUT -p tcp -m state --state NEW --dport 443 -j ACCEPT
```

NTP，即网络时间协议（Network Time Protocol），被允许在本地网络内部使用。这里我们要确保在目的地和源端口上进行筛选。可以这样做的理由是，我们知道这个 UDP 连接与每台主机通信所使用的端口：123。

```
-A Firewall-eth1-INPUT -s 192.168.0.0/24 -p udp -m state --state NEW --sport 123 --dport 123 -j ACCEPT
```

最后，在已添加的规则清单中有 IMAP 规则。IMAP 是一个邮件协议，我们将在稍后的第 10 章介绍，这里只允许内部网络访问它。

```
-A Firewall-eth1-INPUT -s 192.168.0.0/24 -p tcp -m state --state NEW --dport 143 -j ACCEPT
```

这样就创建了使 `headoffice.example.com` 主机活动并运行的必要规则。现在让我们配置 `gateway.example.com`。

gateway.example.com 防火墙

那么，为了使主机变为网关防火墙以使它能够区分来自 `ppp0` 或者 `eth1` 上所有种类的包，然后通过合适的接口路由我们的包，我们需要往主机上添加什么呢？我们需要再添加一些链和 `nat` 表。

补充知识：各种各样的协议

Netfilter 或 iptables 基于 TCP/IP 协议栈里的不同的协议进行筛选。筛选可以基于协议，如 TCP、UDP、AH、IPsec、PPPoE、STP 和许多其他协议。

在 `/etc/protocols` 里你可以得到所有可能的协议与它们的协议号组成的不错的清单。在 iptables 里，你声明规则时可以指定协议名，如 `udp` 或协议号，如 17（UDP 的号）。下面的例子在 UDP 包里筛选。

```
iptables -A INPUT -p 17 -m state --state NEW --dport 53 -j ACCEPT
```

如果没有指定协议，它将默认匹配所有协议。你还可以使用 `!` 定义规则来排除一个协议，如下所示。

```
iptables -A INPUT -p ! udp -m state --state NEW -j ACCEPT
```

这个规则接受除了使用 UDP 协议的包之外所有处于 NEW 状态的协议包。下面的网址是一个不错的有关网络协议的参考文献：<http://www.protocols.com/pbook/tcpip1.htm>。

配置我们的规则：nat 表

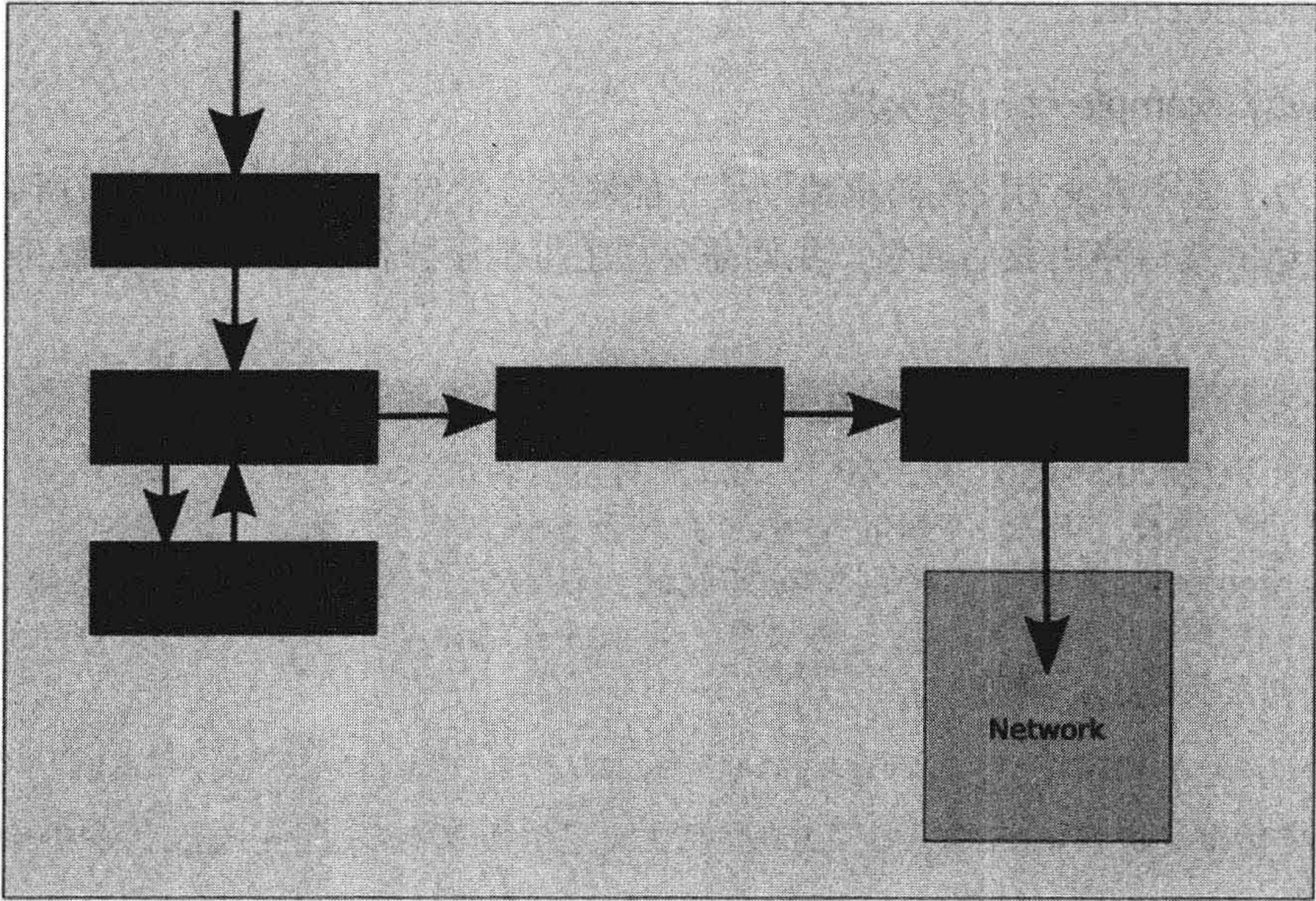
我们现在从 filter 表中断一会，看看所需的 `nat` 表中的配置。`nat` 表的定义已经早在本章的“表”一节中说明过了。`nat` 表处理 IP 伪装和 DNAT（端口映射）规则。我们需要配置下面的内置链：PREROUTING 和 POSTROUTING。`nat` 表用来路由和改变从 `ppp0` 接口进出的流量。现在我们把这些行添加到文件中。


```
*nat
:PREROUTING ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A PREROUTING -i ppp0 -p tcp -m tcp --dport 25 -j DNAT --to-destination 192.168.0.1
-A PREROUTING -i ppp0 -p tcp -m tcp --dport 80 -j DNAT --to-destination 192.168.0.1
-A POSTROUTING -o ppp0 -j MASQUERADE
COMMIT
```

只需几个规则就可让它工作。Netfilter 加载 *nat 和 COMMIT 行之间的所有规则，并顺次读取它们，就像它在 filter 表中所做的那样。要使 Netfilter 能使用 nat，你需要把 iptable_nat 加载到内核。通过发出下面的命令你可以检查是否已经加载。

```
$ sudo /sbin/lsmod |grep table
ip_tables                17029 2 iptable_nat,iptable_filter
```

让我们分析一下包通过已配置 NAT 的主机的过程。图 6-31 绘制了一个包可能采取的与 nat 表相关的路径。



Network: 网络

图 6-31 与 nat 表相关的包流程

包在接口上进来并且会遇到 PREROUTING 链。在这里我们可以执行 DNAT 操作（端口映射）。一旦通过了这个链，这些包就会按照它们的最终地址路由。它们会被传送给 filter 表，通过 FORWARD 链转发或者传送给 INPUT 链。然后包通过 nat 表里的 OUTPUT 链被传回，在那里由防火墙产生的包可能被改变（如果需要的话）。POSTROUTING 链的用途是做包伪装以使它们看起来来自公共 IP 地址。这个地址一定是外部接口的基本 IP 地址。如果你想在防火墙上为外部连接取别名，那么就必须使用 SNAT。

从上述 nat 表的例子你还可以看到已创建的 3 个链（这些都是 nat 表的默认内置链）。你可以使用:chain-name 表示一个链。这样，进入主机的流量所发生的事情就可以分解成几部分。

在第一部分，所有的流量被导向 PREROUTING 链并被接受，如 PREROUTING ACCEPT 所示。接下来，从接口 `ppp0` 进来（这里写作 `-i ppp0`）且使用 TCP 协议（如 `-p tcp` 所示）的所有流量被标记。对于任何使用协议 TCP（这里重新写作 `-m tcp`）且目的地为 `ppp0` 接口上的端口 25（如 `--dport 25` 所指）的包，我们都希望发送给 DNAT 目标，它会把这些包转发到 IP 地址 192.168.0.1（如 `-j DNAT --to-destination 192.168.0.1` 所指）。我们以同样的方式处理在端口 80 进入的流量。

接下来的一行处理所有通过 `ppp0` 接口离开主机的流量。包被路由之后，它们通过 POSTROUTING 朝着网络做最后的冲刺。

```
-A POSTROUTING -o ppp0 -j MASQUERADE
```

这是 `nat` 链的最后一行。它的意思是，当流量离开 `ppp0` 接口（如 `-o ppp0` 所指）时，把包发送给扩展目标 MASQUERADE（如 `-j MASQUERADE` 所指）。这是在做什么？如前所述，这使得所有流量离开 `ppp0` 并被放到一个 `nat` 表中。这个表记录走出这个接口的流量。它记录原始的源 IP 和目的 IP。此时，防火墙重写了包以使它的源 IP 地址就是公共 IP 地址，并把此包发给它的目的主机。回应连接的包通过防火墙返回。对于输入包，使用 `nat` 表以便匹配给发起主机。

配置我们的规则：重访 filter 表

现在有了基本的 filter 规则并且配置了 `nat` 表里的链，那么我们要重访 filter 表，以确保流量在两者之间正确地流动。包要从一个接口到另一个接口地寻找它们的路径。在一个接口进入、在另一个接口出去的流量需要使用 FORWARD 链。

我们打算为 filter 表创建另外一组链。这些链处理所有进出 `ppp0` 或者说公共接口的流量，称之为 `Firewall-ppp0-chain name`。

```
:Firewall-ppp0-INPUT
:Firewall-ppp0-FORWARD
:Firewall-ppp0-OUTPUT
```

`Firewall-ppp0-INPUT` 链要处理所有从 `ppp0` 接口进入防火墙的流量。像下面这样导引从 `ppp0` 进入的流量。

```
-A INPUT -i ppp0 -j Firewall-ppp0-INPUT
```

这个链与目的地为防火墙本身的公共（Internet）流量有关。我们希望用与转发给其他主机的包不同的方式处理这个流量，因此不想把它与 FORWARD 规则结合。相反，我们创建如下规则。

```
-A FORWARD -i ppp0 -o eth1 -j Firewall-ppp0-FORWARD
```

在这里我们要为从公共接口进入、通过私有接口 `eth1` 转发的包制定规则。最后我们希望用与内置 OUTPUT 链不同的方式处理防火墙发出的流量。

```
-A OUTPUT -o ppp0 -j Firewall-ppp0-OUTPUT
```

现在让我们看一看 filter 表的完整规则组。可以看到，我们已经向用户自定义的链添加了几个规则，以允许网络上的期望流量。


```
*filter
:INPUT DROP
:FORWARD DROP
:OUTPUT DROP

:Firewall-1-INPUT
:Firewall-ppp0-FORWARD
:Firewall-ppp0-INPUT
:Firewall-ppp0-OUTPUT

-A INPUT -i eth1 -j Firewall-eth1-INPUT
-A INPUT -i ppp0 -j Firewall-ppp0-INPUT
-A FORWARD -i eth1 -o eth1 -j Firewall-eth1-INPUT
-A FORWARD -i eth1 -o ppp0 -j Firewall-ppp0-INPUT
-A FORWARD -i ppp0 -o eth1 -j Firewall-ppp0-FORWARD
-A OUTPUT -o ppp0 -j Firewall-ppp0-OUTPUT
-A OUTPUT -s 192.168.1.0/255.255.255.0 -d 192.168.1.0/255.255.255.0 -j ACCEPT

-A Firewall-eth1-INPUT -i lo -j ACCEPT
-A Firewall-eth1-INPUT -p icmp -m icmp --icmp-type any ➡
-m limit --limit 3/s -j ACCEPT
-A Firewall-eth1-INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A Firewall-eth1-INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A Firewall-eth1-INPUT -p tcp -m state --state NEW -m tcp --dport 53 -j ACCEPT
-A Firewall-eth1-INPUT -p udp -m state --state NEW -m udp --dport 53 -j ACCEPT
-A Firewall-eth1-INPUT -p tcp -j LOG --log-prefix "eth1_in_tcp_REJECT "
-A Firewall-eth1-INPUT -j REJECT --reject-with icmp-host-prohibited

-A Firewall-ppp0-INPUT -p icmp -m icmp --icmp-type any ➡
-m limit --limit 5/s -j ACCEPT
-A Firewall-ppp0-INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A Firewall-ppp0-INPUT -p tcp -m state --state NEW -m tcp --dport 53 -j ACCEPT
-A Firewall-ppp0-INPUT -p udp -m state --state NEW -m udp --dport 53 -j ACCEPT
-A Firewall-ppp0-INPUT -m limit --limit 5/s -j LOG --log-prefix "ppp0_in_REJECT "
-A Firewall-ppp0-INPUT -j REJECT --reject-with icmp-host-prohibited

-A Firewall-ppp0-FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
-A Firewall-ppp0-FORWARD -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A Firewall-ppp0-FORWARD -p tcp -m state --state NEW -m tcp --dport 25 -j ACCEPT
-A Firewall-ppp0-FORWARD -p tcp -m state --state NEW -m tcp --dport 53 -j ACCEPT
-A Firewall-ppp0-FORWARD -p udp -m state --state NEW -m udp --dport 53 -j ACCEPT
-A Firewall-ppp0-FORWARD -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
-A Firewall-ppp0-FORWARD -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
-A Firewall-ppp0-FORWARD -p udp -m state --state NEW ➡
-m udp --sport 123 --dport 123 -j ACCEPT
-A Firewall-ppp0-FORWARD -j REJECT --reject-with icmp-port-unreachable

-A Firewall-ppp0-OUTPUT -p icmp -m icmp --icmp-type any -j ACCEPT
-A Firewall-ppp0-OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A Firewall-ppp0-OUTPUT -p tcp -m state --state NEW -m tcp --dport 53 -j ACCEPT
-A Firewall-ppp0-OUTPUT -p udp -m state --state NEW -m udp --dport 53 -j ACCEPT
-A Firewall-ppp0-OUTPUT -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
-A Firewall-ppp0-OUTPUT -j DROP
COMMIT
```


我们把它分解成小块以便于理解。首先看看对 Firewall-eth1-INPUT 链做了什么。

```
-A Firewall-eth1-INPUT -i lo -j ACCEPT
-A Firewall-eth1-INPUT -p icmp -m icmp --icmp-type any ➡
-m limit --limit 3/s -j ACCEPT
-A Firewall-eth1-INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A Firewall-eth1-INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A Firewall-eth1-INPUT -p tcp -m state --state NEW -m tcp --dport 53 -j ACCEPT
-A Firewall-eth1-INPUT -p udp -m state --state NEW -m udp --dport 53 -j ACCEPT
-A Firewall-eth1-INPUT -p tcp -m limit --limit 5/s ➡
-j LOG --log-prefix "eth1_in_tcp_REJECT "
-A Firewall-eth1-INPUT -j REJECT --reject-with icmp-host-prohibited
```

它对从 eth1 转发或从私有网络进入到防火墙的流量起作用。如前所述，我们允许回送地址做任何它喜欢的连接，允许 ICMP 流量和任何 ESTABLISHED 或 RELATED 状态的连接，还允许向 SSH 服务器和 DNS 服务器发起的连接。前面已经研究了除-j LOG 目标之外所有的这些信息。对于-j LOG，如下所示。

```
-A Firewall-eth1-INPUT -p tcp -m limit --limit 5/s ➡
-j LOG --log-prefix "eth1_in_tcp_REJECT "
```

它的语法是：iptables -t table chain parameters -j LOG --log-prefix some string。被-p tcp 捕获而没有与任何规则匹配的 TCP 流量会被系统日志后台程序（它的文件可在/var/log/messages 或/var/log/syslog 里找到）记录。在本例中，它们以 eth1_in_tcp_REJECT 为前缀。

这里给出 Red Hat 主机上/var/log/messages 中所记录的一个消息例子。

```
Jan 17 00:02:23 localhost kernel: eth1_in_tcp_REJECT IN=eth1 OUT= ➡
MAC=00:0b:cd:49:c0:7f:00:02:2d:87:6e:0c:08:00 SRC=192.168.0.1 ➡
DST=192.168.0.254 LEN=52 TOS=0x00 PREC=0x00 TTL=128 ID=2178 DF PROTO=TCP ➡
SPT=1556 DPT=445 WINDOW=16384 RES=0x00 SYN URGP=0
```

日期戳 Jan 17 00:02:23 之后是主机名 localhost 和产生该记录的程序 kernel。现在你可以看到记录前缀字符串 eth1_in_tcp_REJECT。紧接着的是拒绝所有包的规则，因此正在记录的是即将被拒绝的流量。你接下来看到的是流量的细节。IN 和 OUT 表示流量来自的地方。之后你可以看到所涉及主机的 MAC 地址和它们的 SRC 和 DST（源和目的地）IP 地址。其他各种参数也被记录，包括源端口 SPT 和目的地端口 DPT。

这个规则组的最后是拒绝所有未匹配流量的 REJECT 规则。前面已经分析了这个规则。

现在研究一下从 ppp0 公共接口进入目的地为防火墙本身的流量的处理方式。这个流量通过 INPUT 链被路由到 Firewall-ppp0-INPUT 链。

```
-A Firewall-ppp0-INPUT -p icmp -m icmp --icmp-type any -m limit ➡
--limit 5/s -j ACCEPT
-A Firewall-ppp0-INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A Firewall-ppp0-INPUT -p tcp -m state --state NEW -m tcp --dport 53 -j ACCEPT
-A Firewall-ppp0-INPUT -p udp -m state --state NEW -m udp --dport 53 -j ACCEPT
-A Firewall-ppp0-INPUT -m limit --limit 5/s -j LOG --log-prefix "ppp0_in_REJECT "
-A Firewall-ppp0-INPUT -j REJECT --reject-with icmp-host-prohibited
```

这里允许 ICMP 连接，并且限制一秒接受的数量（-m limit -limit 5/s）。通过限制人们发起 ICMP 连接的速度，有助于提高对 SYN 洪水攻击的抵抗力。我们接受处于建立或相关状态

的连接。在防火墙上开放的端口只有 UDP 和 TCP 协议的 DNS 端口。用两个 DNS 域名服务器以防其中一个离线的情况并不少见，防火墙就担当这些域名服务器中的一个。可以看到我们还限制对被拒绝包的记录，以使日志不被假信息淹没。假信息同样可以导致系统崩溃，因为日志会填满磁盘。

补充知识：各类攻击

人们可以利用不同类型的攻击使网络崩溃。一些攻击以网络周边为目标，利用假连接尝试直到它中断或停止回应；这就是有名的服务拒绝（Denial of Service, DoS）攻击，例如 SYN 洪水攻击。其他的攻击利用正在运行的服务（像邮件或网络服务器）绕过周边安全防护措施，从网络内部发起攻击（例如病毒攻击、木马攻击、脚本注入攻击）。

保持网络安全是一个持续不断的过程，做下面几件事可以使风险最小化。

- 跟踪主机安全警告的邮寄名单。
- 保持主机得到最新安全补丁程序的更新。
- 关闭不必要的端口和服务。
- 了解日志并学习检测看起来奇怪的东西。
- 至少每 6 个月审查一下安全性，确保它仍然工作正常。

只要连接到 Internet，就没有 100% 的安全性。但是你可以采取措施确保网络没有众所周知的弱点。对于脚本小子来说，不需要花费很长时间就可找到你的主机并尝试一些可预防的普通攻击方式。

该转到另一个规则组了，该规则组决定什么流量可以通过接口从 ppp0 转发到 eth1。这些规则必须存在，以使任何到 Internet 的连接成为可能。当尝试在 Internet 上发起一个连接或套接字（例如去一个网站）时，首先发送一个 SYN 包（带顺序号的同步包）。它告诉网络服务器希望开启一个套接字。它会发回一个 SYN, ACK 包，iptables 把该包视为处于 NEW 状态。只有接收到顺序号增加 1 的进一步 SYN 包从而建立连接之后，状态才是 ESTABLISHED。因此要允许目的地为网络内某处的 NEW 状态的包进入防火墙接口，这样才可以建立连接。

```
-A Firewall-ppp0-FORWARD -p icmp -m icmp --icmp-type any ➡  
-m limit --limit 3/s -j ACCEPT  
-A Firewall-ppp0-FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT  
-A Firewall-ppp0-FORWARD -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT  
-A Firewall-ppp0-FORWARD -p tcp -m state --state NEW -m tcp --dport 25 -j ACCEPT  
-A Firewall-ppp0-FORWARD -p tcp -m state --state NEW -m tcp --dport 53 -j ACCEPT  
-A Firewall-ppp0-FORWARD -p udp -m state --state NEW -m udp --dport 53 -j ACCEPT  
-A Firewall-ppp0-FORWARD -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT  
-A Firewall-ppp0-FORWARD -p tcp -m state --state NEW -m tcp --dport 443 -j ACCEPT  
-A Firewall-ppp0-FORWARD -p udp -m state --state NEW ➡  
-m udp --sport 123 --dport 123 -j ACCEPT  
-A Firewall-ppp0-FORWARD -j REJECT --reject-with icmp-port-unreachable
```

正是考虑到这个理由我们才允许端口 25、53、80 和 443 上 NEW 状态的连接——所有这些端口和希望进入主机 192.168.0.1 的流量相关。要建立连接，邮件需要端口 25，DNS 需要端口 53，而网站服务需要端口 80 和 443。我们希望允许内部网络上的其他主机从端口 123 连接到邮件、网站和时间服务器以及从端口 22 连接到 SSH 服务器，因此也要接受这些连接。

最后，让 Firewall-ppp0-OUTPUT 链处理所有从防火墙到 Internet 的输出流量。你可以看到这些规则都很简单。


```
-A Firewall-ppp0-OUTPUT -p icmp -m icmp --icmp-type any -j ACCEPT
-A Firewall-ppp0-OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A Firewall-ppp0-OUTPUT -p tcp -m state --state NEW -m tcp --dport 53 -j ACCEPT
-A Firewall-ppp0-OUTPUT -p udp -m state --state NEW -m udp --dport 53 -j ACCEPT
-A Firewall-ppp0-OUTPUT -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
-A Firewall-ppp0-OUTPUT -j DROP
```

我们允许输出 ICMP 流量和任何 ESTABLISHED、RELATED 状态的包。我们不限自己的 ICMP 连接输出速度，还允许防火墙主机与任何 DNS 域名服务器连接，并且希望允许与两个网络服务器通信。为此，我们分别指定--dport 53 和--dport 80。允许端口 80 上的出站连接是为了让主机可以接收软件包更新。如果希望把它进一步锁定，我们可以通过--destination 参数指定防火墙可以连接到的特定主机，就像这样：--destination myrepo.example.com -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT。它只允许到 myrepo.example.com 的出站连接，这里的 myrepo.example.com 是在我们的控制之下的一个 APT 或者 Yum 软件资料库（第 7 章会详细介绍 Yum 和 APT）。

6.5.2 其他防火墙配置工具

了解了所有这些内容之后，你可以看出 Netfilter 防火墙和它们的规则可以变得十分复杂。那么有帮助创建防火墙配置的工具吗？

是的，有。Red Hat 主机带了一个工具，在第一次安装主机时可以看到它。这是一个简单的工具，不能制定 filter 表之外的防火墙规则。通过发出下面的命令你可以访问它。

```
$ sudo system-config-security
```

至于这个工具的使用，Red Hat 有一些不错的文档，请参阅 <http://www.redhat.com/docs/manuals/enterprise/RHEL-4-Manual/sysadmin-guide/ch-basic-firewall.html>。

早前已经看到了它所提供的这些规则组，我们推荐检查一下某些 Red Hat 默认规则并移除一些不必要的东西（比如许可 IPP 和 IPsec 的规则）。

Ubuntu 也有一个从命令行处理防火墙规则的工具。它叫 ufw，来自 ufw 软件包。

提示：第 7 章将详细讲述软件包的安装。

ufw 的可用文档请参阅 <https://wiki.ubuntu.com/UbuntuFirewall>。这个软件包使用类似下面这样的命令。

```
$ sudo ufw allow 80/tcp
```

这个命令打开了通过 TCP 对端口 80 的访问权限。命令可以存储在脚本里，并在主机启动时运行。

至于更复杂的情况，我们建议研究另一个叫作 shorewall 的配置工具。它可以从 <http://shorewall.net/> 下载，在这个网址还可以查看这个工具的文档。shorewall 在 Ubuntu 和 Fedora 上也可用，在线软件资料库里的软件包名为 shorewall。

shorewall 涉及了一些复杂的防火墙规则，涵盖 DNAT、SNAT、IPsec、双 WAN 接口配置

和路由以及更多的内容。它写了很干净的 iptables 规则，意思就是没有默认添加任何内容，就像 Red Hat 的 system-config-security 工具有时所做的那样。

■注：这种软件的另一个例子叫 Firestarter (<http://www.fs-security.com/>)。对于单机的 iptables 配置和强力攻击检测，还可以到 <http://www.rfxnetworks.com/apf.php> 研究高级策略防火墙 (Advanced Policy Firewall, APF) 和强力攻击检测 (Brute Force Detection, BFD) 的用途 (对网关防火墙主机不合适)。

还有些 Linux 发行版就是为防火墙构建的。IPCop 就是这种类型的发行版。它可以从 <http://www.ipcop.org/index.php> 下载和安装。它有一个基于网络的图形配置工具，是为 SOHO 业务而设计的单独的防火墙。这不是一个 Red Hat 或 Ubuntu 发行版。它有自己的安装和更新软件包的方式，而这已超出本书的范畴。

6.6 TCP Wrappers

最后，在其他保护主机的方式中，有一种是使用 TCP Wrappers。如果网络服务被编译为支持 TCP Wrappers，你就可以使用 TCP Wrappers 进一步保护主机上的服务。TCP Wrappers 通过 /etc/hosts.allow 和 /etc/hosts.deny 文件里的一系列规定，控制对运行在主机上的后台程序而不是端口的访问。

hosts.allow 里的规则比 hosts.deny 里的规则优先级高。如果你打算使用 TCP Wrappers，在 hosts.deny 里设置下面的命令是个好主意。

```
ALL: ALL
```

这将把对所有服务的默认动作设置为拒绝，如果没有在 /etc/hosts.allow 文件里指定的话。规则的阅读顺序是从顶到底。

你接下来可以添加网络服务。例如，对于范例网络，通过设置下面的命令可以允许网络服务。

```
ALL: localhost ACCEPT
sshd: .example.com EXCEPT .baddomain.com
```

这些设置首先允许接受 localhost 的连接。许多服务需要连接到运行在回送 (localhost) 接口上的服务，因此这些要接受。接下来，我们允许 example.com 网络上的主机连接到 SSH 后台程序。这里还明确拒绝了 baddomain.com。有关配置 TCP Wrappers 的更多信息，请看 hosts_options 或 hosts.allow 和 hosts.deny 的 man 页。

6.7 小 结

现在我们为堡垒主机完成了一个基本的防火墙。在此过程中，我们说明了可以用 filter

表只保护网络上的单机。现在你知道了如何利用防火墙上的 NAT 使私有网络 IP 地址伪装成单个公共 IP。

本章涵盖的内容如下所示。

- 配置主机上的接口。
- 配置一个 PPP 连接。
- 配置堡垒防火墙。
- 使用防火墙上的 NAT。
- 端口映射连接。
- 保护主机。
- 使用 TCP Wrappers 保护后台程序。

下一章将讲述如何管理主机上的软件包。在那一章你将学到如何安装、移除和更新主机上的软件。

第 7 章 软件包管理

在第 2 章，我们安装了第一台 Linux 主机。正如在那章介绍的那样，主机上能够安装各种各样的应用程序，既可以选择单一安装也可以选择安装所有相关的应用程序和工具。但这并不是安装和管理应用程序的全部内容。一旦主机安装完毕，你常常需要添加、升级应用程序以及安装相应的补丁程序，有时还需要移除应用程序。本章将讲述如何在 Red Hat Enterprise Linux 和 Ubuntu 上完成这些工作。

在 Linux 发行版中，这种对应用程序的管理被称作软件包管理。这是因为 Linux 主机上绝大多数应用程序都是以软件包的形式出现的，这样，安装和移除应用程序就很简单。软件包通常包括所有的二进制文件、配置文件以及安装应用程序所需的其他支持文件，软件包还知晓上述文件安装的具体位置以及安装该软件包时所需的其他软件包。比较而言，Linux 软件包非常类似于执行 Windows 安装或者配置的可执行文件。

软件包安装通常由一个应用程序控制中心（即软件包管理器）来完成。大多数的 Linux 发行版自带一个软件包管理器。它通常包括一套用于管理软件包的命令行工具和图形化工具，以及一个小型数据库（用于记录安装了哪些软件包）。

在本章结束时，你将会对软件包有一个全面的认识，知道如何安装和移除它们、如何找到需要的软件包以及如何从源代码中安装软件。我们将会通过 Red Hat 和 Ubuntu 发行版中的命令行工具和图形化工具演示以上所有的工作。

■注：在“软件包管理介绍”一节之后，本章将分为“Red Hat 上的软件包管理”、“Ubuntu 上的软件包管理”、“从源代码安装软件”三节。读者可根据所安装的 Linux 发行版选择学习上述章节。我们还推荐阅读“从源代码安装软件”这一节，即便不会经常用到（我们推荐始终通过软件包管理应用程序），但是这一节的内容非常有用。

7.1 软件包管理介绍

不同的发行版对它们的软件有不同的打包方式。例如，Red Hat Linux 和 CentOS 使用 RPM（Red Hat Package Management）包格式，而 Ubuntu 使用 deb（Ubuntu 最初基于的 Debian 的简称）格式。正因为包格式不同，所以管理它们的工具也不同。

这些软件包类型每一个都使用不同的工具安装和管理软件包。在使用 RPM 包的系统上，基本的软件包管理工具叫作 rpm，而在使用 deb 包的系统上，它叫作 dpkg。二者都是极强大的应用程序，可用来处理系统上的软件包。除了这些基本的软件包管理器，还有一些提供额外功能的应用程序，如基于网络的更新程序、软件包搜索程序和 GUI 工具。

■注：存在不同的软件包格式和软件包管理器，这也许看起来奇怪——毕竟它们都是 Linux 发行版——这主要是历史原因造成的。起初创建这些发行版时，开发者们对软件包系统应该如何工作没有达成一致意见，于是就各建各的。多年来，这些软件包系统的开发持续进行，到现在就有了多种成熟的软件包系统和软件包格式。当然，如果只使用一个发行版，你就只需学习一种类型的软件包管理。

尽管所有的 Linux 发行版都可以包含成千上万的软件包，但是一般来说，这些包都属于以下 3 种主要类别。

- 应用程序包。
- 库包。
- 开发包。

顾名思义，应用程序包包含应用程序。这些应用程序可以是简单的命令行编辑器，也可以是完整的 OpenOffice 生产力组件。

■注：OpenOffice 是与 Microsoft Office 同等功能的开源软件。它包含字处理器、表格处理程序、演示文稿软件以及其他工具。它允许编辑 Microsoft Office 文档，并提供与 Microsoft Office 类似的功能。

库包包含应用程序与操作系统为提供辅助功能所使用的文件。例如，libssl 包提供密码支持文件。与社区书库很类似，在 Linux 库中，应用程序可以寻找那些自己没必要拥有的文件。因为这些库通常由多个应用程序使用，所以明智的做法是把它们放在它们自己的一个包中，而不是每个应用程序都包括每个库的副本。如果一个库包更新了，那么使用这个库的所有应用程序马上自动使用最新的版本。这些包的名字通常以 lib 开头。

开发包包含从源编译所需的源代码和头文件。这些包的名字通常以-dev（在 Ubuntu 上）或-devel（在 Red Hat 上）结尾。大部分库包都有一个伴随的开发包，以方便使用这些库做软件开发。一般地，除非是在开发应用程序，否则你不必安装这些包。但有些应用程序确实要用到它们，如果选择从源编译应用程序并按那种方式安装它们，那么通常就需要开发包来做此事。

因为 Red Hat 与 Ubuntu 所使用的软件包管理工具完全不同，所以分两节介绍它们。我们首先介绍 Red Hat 上的软件包管理工具，然后介绍 Ubuntu 上的工具。

补充知识：什么是软件包？

软件包是为了更容易管理应用程序而设计的。它们一般是根据应用程序的源代码构建的，并且有逻辑设置告诉发行版该应用程序的二进制代码、文件和配置要放到哪里。我们将使用两种类型的软件包：RPM 和 deb 文件。这两种软件包都是包含其他文件的存档文件。那么这些软件包里面是什么？软件包包含数据、元数据，有时还包含一些控制文件。数据是将被安装的文件。控制文件包含有关该软件包的说明信息、用户交互的脚本以及管理安装前或安装后自动化任务的脚本。

7.2 Red Hat 上的软件包管理

在最基本的层面上，基于 Red Hat 的系统上的应用程序管理方式是使用 Red Hat 软件包管理工具，或称 rpm。它被用于如 Red Hat Enterprise Linux、CentOS、Mandriva 和 Fedora Project 这样的发行版。rpm 工具本身是为了在本地系统上安装、处理、查询和删除软件包而设计的。

■提示：通过后缀.rpm 你可以识别 RPM 软件包。每个 RPM 软件包都是根据包含在一个 spec 文件里的信息构建的。spec 文件包含有关每个软件包内容的元数据，并描述该软件包在系统中的安装方式。本章稍后将略述 spec 文件以及如何构建自己的软件包。

rpm 工具提供基本的软件包管理功能，如安装与移除软件包，但不处理其他任务，如从在线软件资料库检索依赖性软件包（即在安装某个软件包时需同时或提前安装的软件包）或软件包的定期自动更新。

■注：因为已经使用 Linux 很长时间了，所以我们可以证明，多亏了灵巧的软件包管理工具，现在管理 Linux 系统容易了许多。在过去，安装一个软件包或更新一个应用程序会把数小时的时间耗费在寻找依赖性软件包（在安装和使用另一个软件包之前必须已在系统上的软件包）上。在这些管理程序出现之前，必须从源构建所有的应用程序，还要处理所引起的冲突。现在就十分简单了。但是，当然了，如果真想这么做，总是可以从源构建的——Linux 是如此强大，可以提供这个选项。本章最后一节会介绍从源构建的方法。

为了提供 rpm 所欠缺的一些功能，大多数 Red Hat 衍生的 Linux 发行版都有一些附加工具。这些工具大多数都是协助从软件资料库（软件包存储的地方；大多数发行版都有许多可用的在线软件资料库）检索软件包并提供这些包以待安装。这类工具包括 Red Hat 的 Red Hat 网络（RHN，它是 Red Hat 的商用更新服务，第2章简要介绍过）、杜克大学的 Yellowdog Updater Modified（Yum）或称 yum，以及 Mandriva 的 urpmi。本节将关注 Red Hat Enterprise Linux 本身所提供的工具，但里面所包含的信息同样有助于其他基于 Red Hat 的发行版，如 CentOS、Fedora 和 Mandriva。

在下面几节里，我们将介绍借助 GUI 界面和命令行进行的软件包安装，以及如何使用 rpm 工具本身管理单个的软件包。我们还将说明如何配置 Red Hat Enterprise Linux 主机让它使用 RHN 以及如何构建自己的 RPM 软件包。

7.2.1 准备开始

在如 Red Hat Enterprise Linux、CentOS 和 Fedora 这类基于 Red Hat 的主机上，最容易的软件包管理方式是通过桌面系统。登入桌面之后，你可以访问相关的 GUI 管理工具。

你可以使用桌面上的两个程序管理主机上的所有软件包，这两个程序可以列出软件包清单，搜索、安装、升级或移除软件包。它们的名字切中主题：软件包管理器（Package Manager）

和软件包更新器（Package Updater）。使用软件包管理器你可以列出、搜索、安装和移除软件包，使用软件包更新器你可以从远程软件资料库更新软件包。

对于 RHEL 主机，你需要有订购才能得到这两个软件包管理工具的全部价值。下一节介绍如何连接到 RHN。

注：如果还没有设置 RHN，你也许想跳到“Red Hat 网络”这一节。在第 2 章安装 Red Hat 主机时也介绍过 RHN 的设置。

对于 Fedora 与 CentOS 主机，没有订购你也可以使用这两个工具。

通过选择“应用程序”→“添加/移除软件”你可以找到软件包管理器程序，通过选择“应用程序”→“系统工具”→“软件包更新工具”你可以找到软件更新器程序。图 7-1 显示了如何从 Red Hat 桌面找到软件更新器。

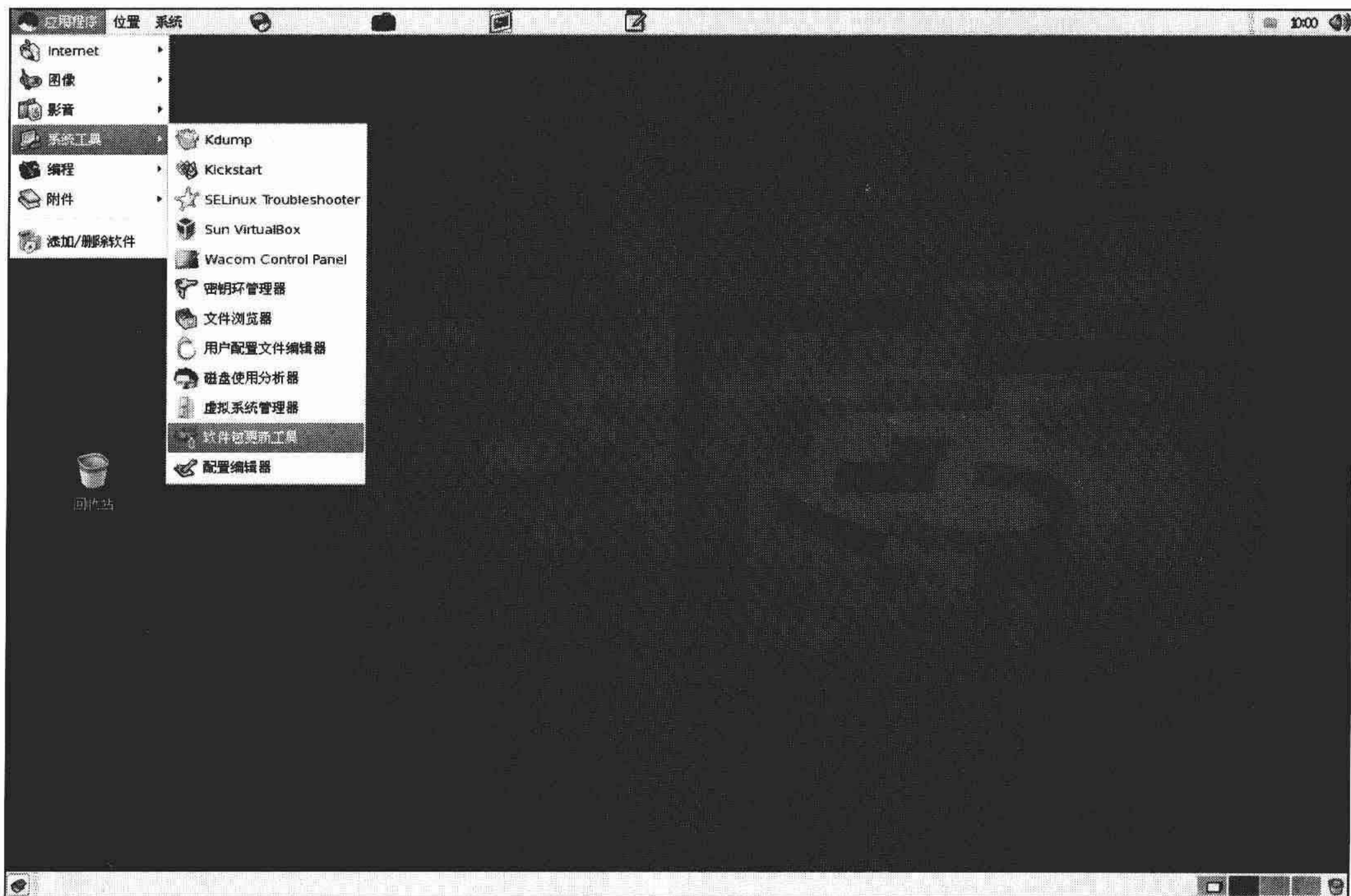


图 7-1 寻找 RHEL 上的软件包管理软件

这些工具非常易用，正如下面的章节中演示的那样。

7.2.2 软件包更新器程序

在登入 Red Hat 或 Fedora 的 GUI 桌面时，你首先会注意到的东西之一是窗口右上角的更新警告图标。如果你确实订购了操作系统，该图标会告知所有适合主机的安全更新。对于我们的 Red Hat 主机，该图标提示有 36 个可用更新，如图 7-2 所示。为了获得这个信息，主机联系了 RHN 或在线更新软件资料库并查询了所有可用更新。这个功能可以用来更新主机上的软件包。



图 7-2 登录后显示的安全警告

单击警告图标会直接进入 Red Hat Enterprise Linux 的软件包更新器，如图 7-3 所示。这个应用程序就是早前通过“应用程序”菜单看到的那个程序。这个程序叫作 pup，它是一个通过 Yum 安装更新的图形前端（“Yellowdog Updater Modified (Yum)”一节会更详细地解释 Yum）。

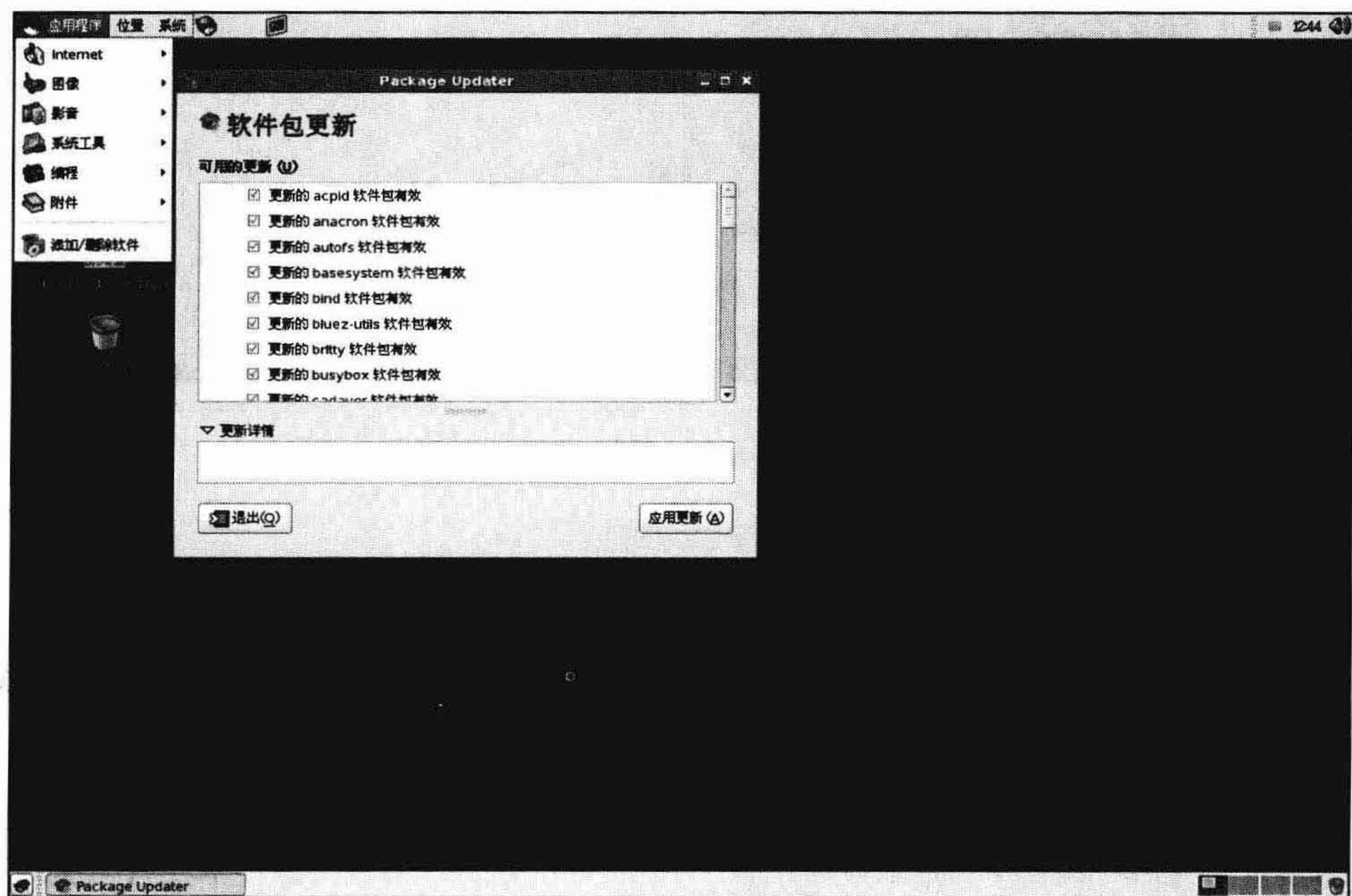


图 7-3 可用更新

为了把所有更新应用到主机，用户必须有 root 特权。通过在得到提示时提供密码并在 `sudo` 命令下运行 `pup` 命令，或者通过以 root 用户身份登入并运行更新，你可以获得 root 特权。如果当前用户没有 root 特权，它会在这里提示输入 root 密码。

从图 7-3 也可以看到可从 RHN 更新的软件包。

如果某个更新需要重启主机，你就可以在这个更新的旁边看到一个环形箭头，如图 7-4 所示。如果更新之后基本上会成为一个新的操作系统，那么这样的更新，如内核更新，就需要一次重启。

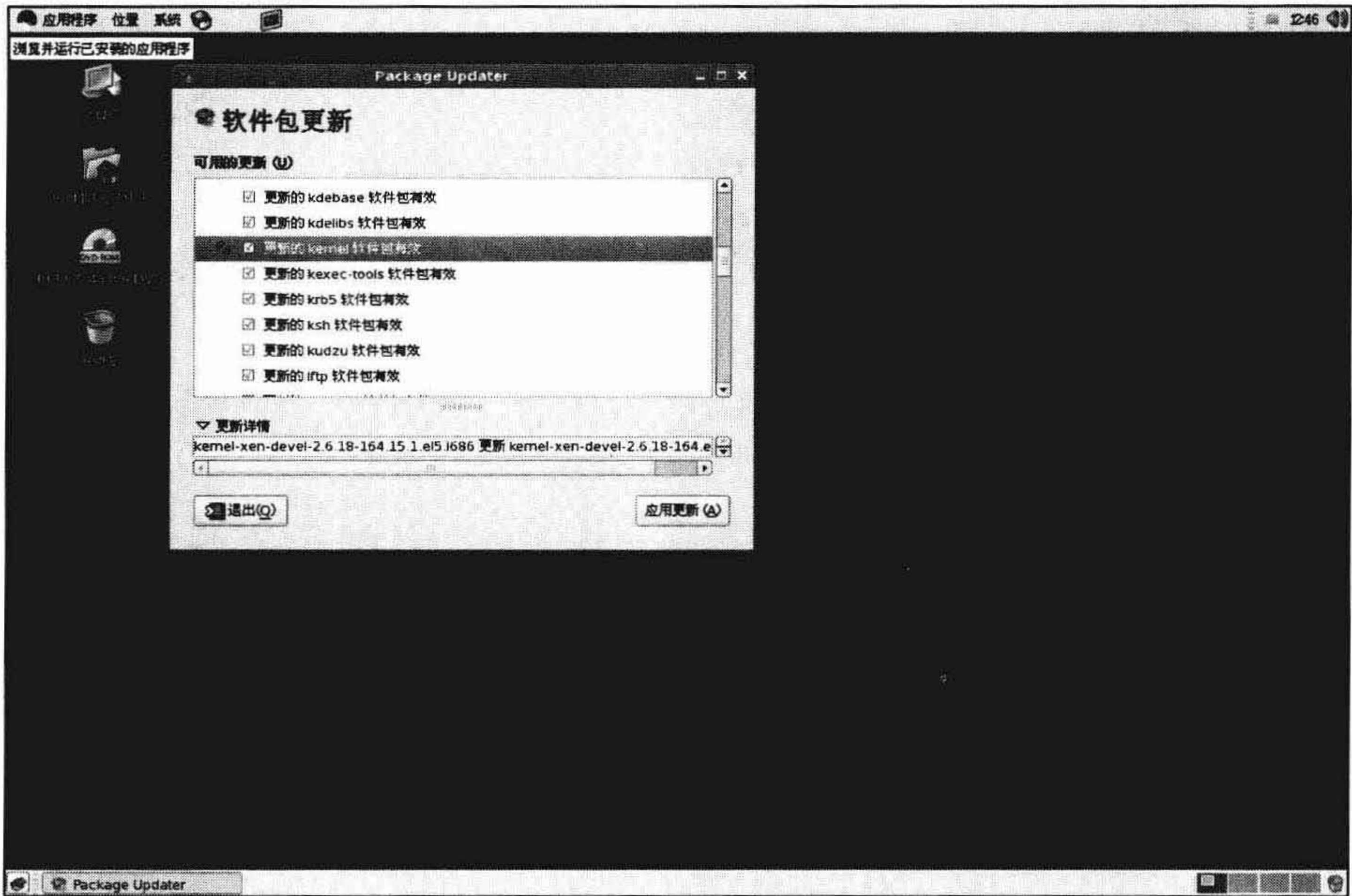


图 7-4 一个需要重启的内核更新

单击“应用更新”按钮会升级所选中的所有软件包。对于不希望升级的软件包，你可以撤销对它们的选择。

了解正在升级的内容是个好习惯，至少应该留意相关的软件包。这样做的原因有两方面。第一，Linux 主机主要由配置文件管理，有时这些配置文件可能由于应用了一个新的更新而被覆盖，从而导致服务不能工作。第二，如果更新的是产品服务器，那么理解有些更新会影响应用程序的行为就很重要。例如，如果正在运行的一个网络服务器提供基于网络的应用程序，那么更新或改变网络服务器的版本或者网络服务可能会中断或者改变应用程序的行为。重要的是阅读并了解伴随更新所发生的改变；这种信息通常在更新详情框中说明。

在第一次更新系统时，它会要求导入 Red Hat 公司的发布密钥。所有 RPM 软件包都可以用一个 PGP/GPG 密码签名，这种做法类似于主机为验证即将安装的软件包的有效性和安全性而做的签名。

提示：PGP/GPG 是一个公钥加密工具。你可以到 http://en.wikipedia.org/wiki/Pretty_Good_Privacy 和 http://en.wikipedia.org/wiki/Public-key_cryptography 上了解 PGP/GPG。

如果正在更新的软件包与 PGP 签名不匹配，那么它可能是恶意软件或者不是来自 RHN。在图 7-5 中，它要求导入 Red Hat 密钥。

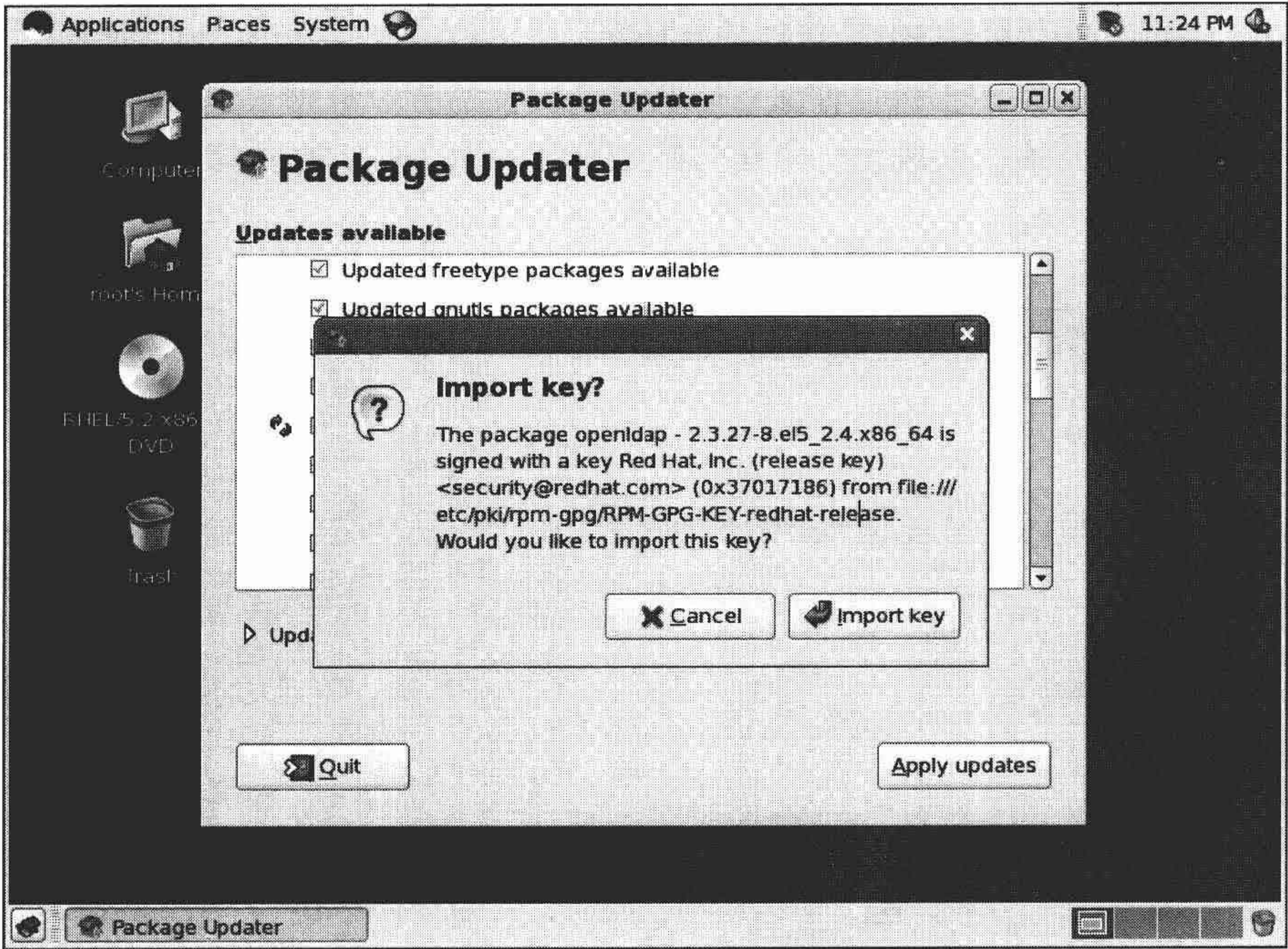


图 7-5 导入 Red Hat 的 PGP 密钥签名

选择导入这个密钥之后，更新进程就会开始下载与安装这些软件包，如图 7-6 所示。

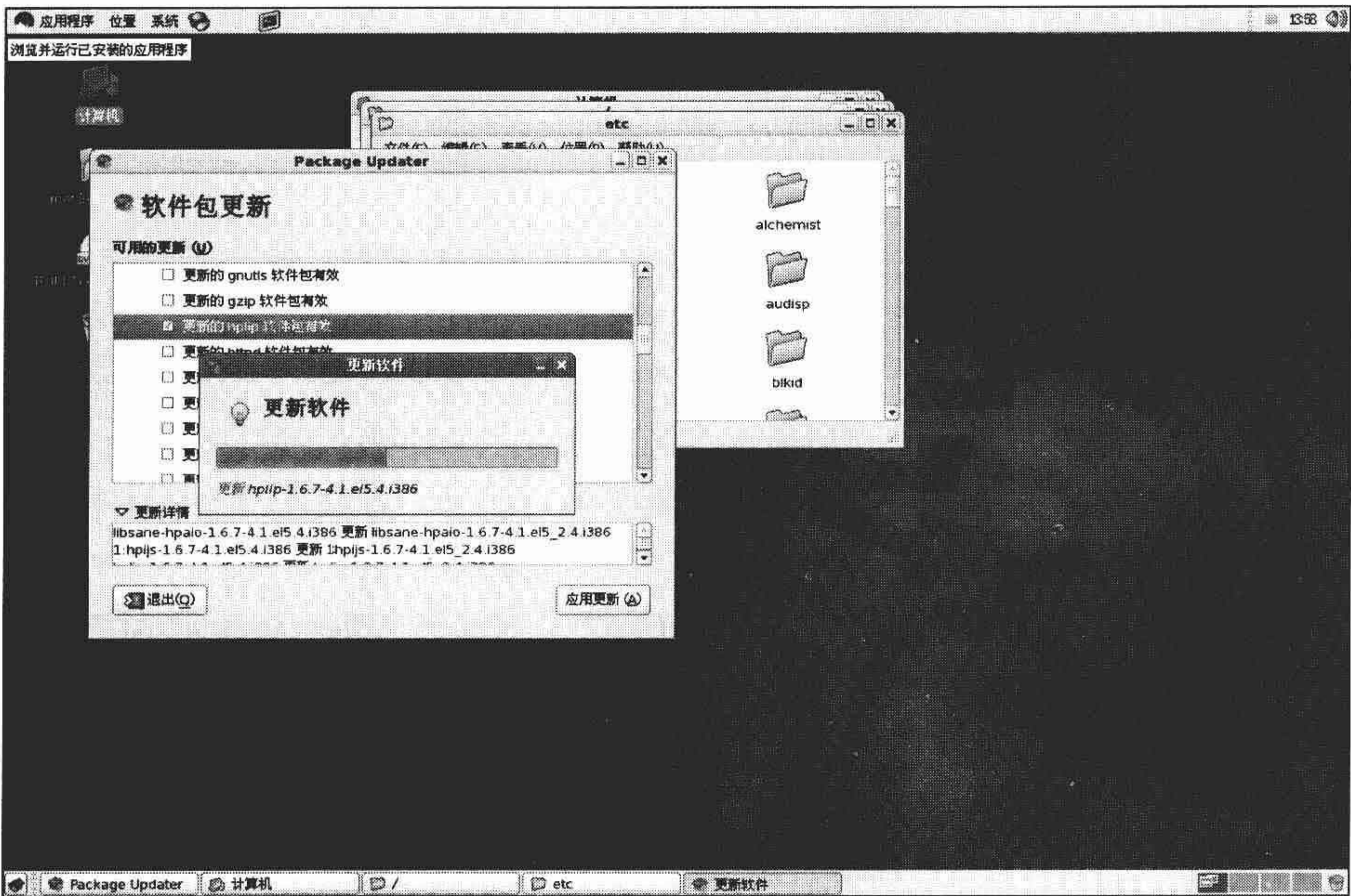


图 7-6 下载与安装软件包

这个进程一完成，它就要求重启。在这种情况下，内核已经升级，如果想使用这个新内核，就需要一次重启。你可以选择不立即重启，而是在一个更方便的时间重启主机。如果你选择重启，主机就会开始关闭并重新启动。

注：如果主机是一个产品服务器，请确保这是一个重启与更新的良机，并通知用户和客户。否则，它们会发现自己在没有警告的情况下就失去了服务器所提供的服务。

如图 7-7 所示，系统正在要求重启。



图 7-7 希望立即重启吗

因为在这个升级过程中安装了一个新内核，所以 `/boot/grub/grub.conf` 文件被重写，新内核被设置为默认值。在主机重启时，它会立刻用新内核启动。

提示：第 2 章和第 5 章介绍了 GRUB 和启动过程。

GRUB 菜单现在提供了两个可供启动进入的内核。正如所讲的那样，新安装的内核将是默认选项。从图 7-8 可以看到内核选项。

注：有些情况下，新内核可能不按预期的方式运转。由于种种原因，有时硬件停止工作或者应用程序出问题。你可以使用 GRUB 菜单启动进入“已知运转良好”的老内核，直到这些问题得到修正为止。

如果在超时时间内没有选择备选内核，主机就会启动进入默认内核。按任意键可以中断超时倒计时，然后使用上下箭头与回车键从菜单选择希望进入的内核。现在主机就带着所有这些新安装并已应用的软件包更新启动。

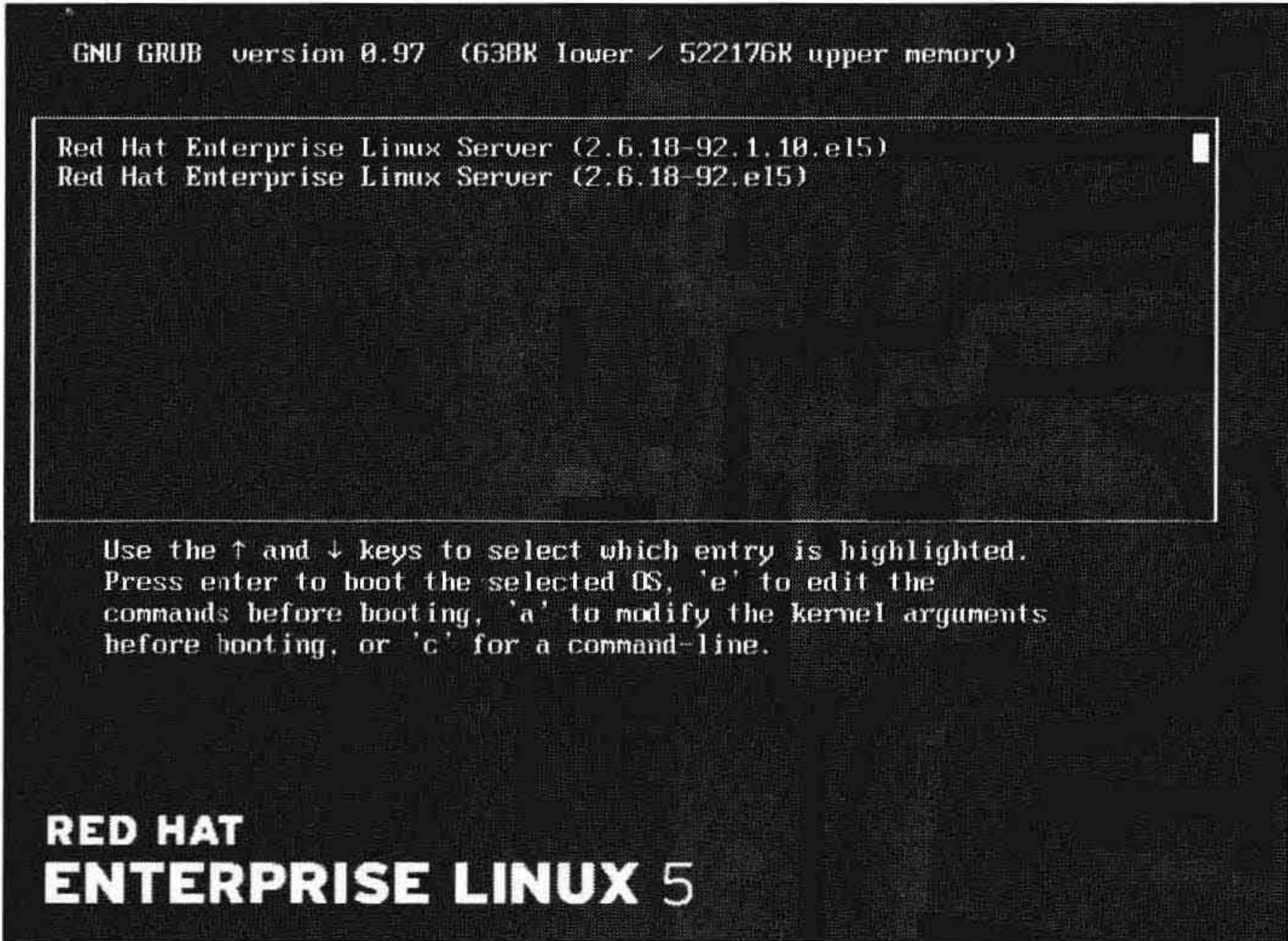


图 7-8 从 GRUB 菜单选择一个内核

7.2.3 软件包管理器程序

现在要看看如何使用菜单“应用程序”→“添加/移除软件”下的软件包管理器程序添加和移除软件。

这个菜单项所运行的应用程序叫作 `pirut`，与相对应的 `pup` 类似，它是 RHEL 上安装、移除、更新、搜索和查看软件包的 Yum 图形前端。它也需要 `root` 特权，如果没有的话，它会提示输入 `root` 用户的密码。

如图 7-9 所示，打开软件包管理器，定位到“列表”标签，并选中“可用的软件包”单

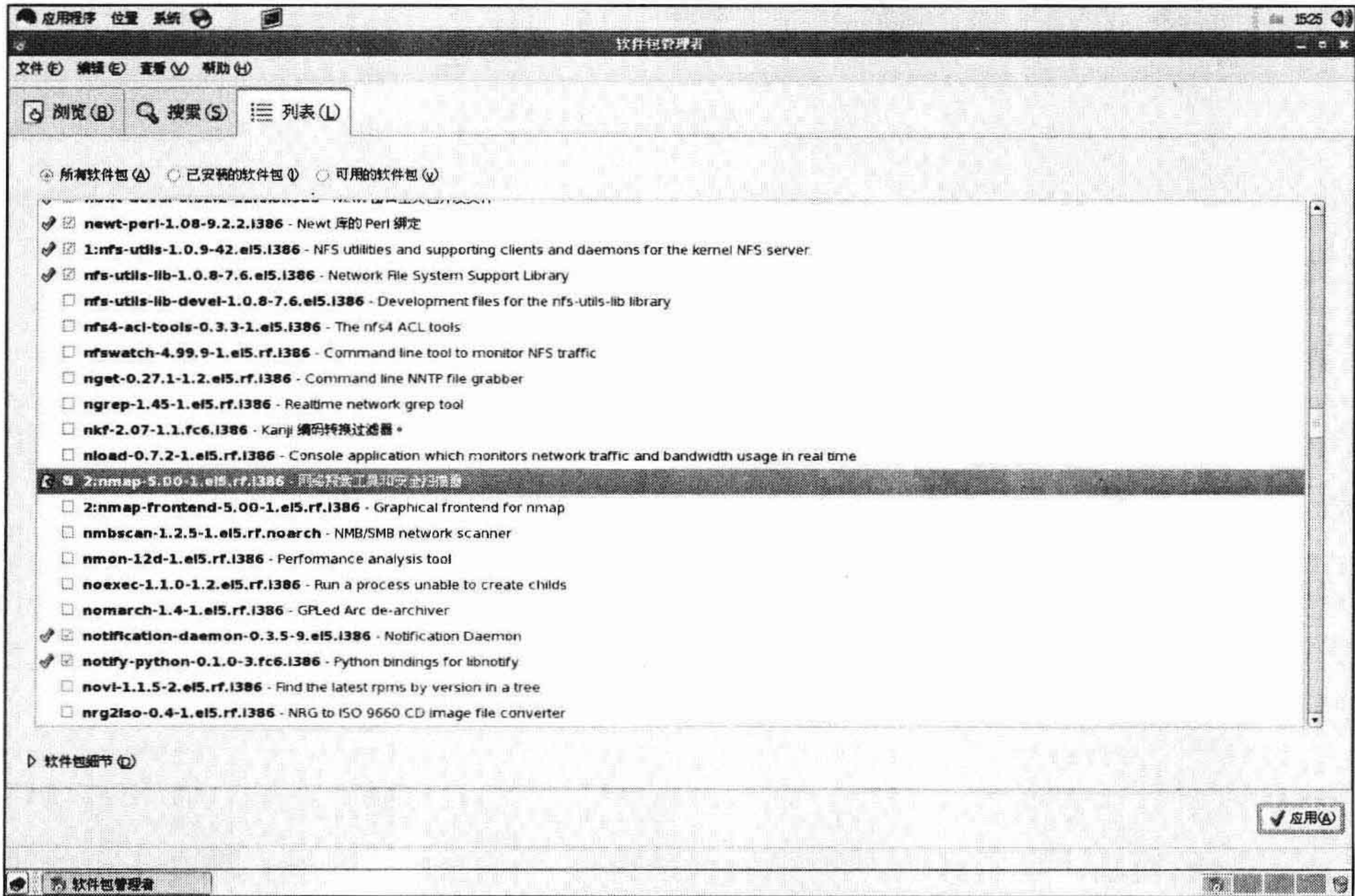


图 7-9 用软件包管理器添加 nmap 软件包

选按钮以缩小待安装的可用软件包列表。我们选中了 **nmap** 软件包，如它旁边选择框中的对勾所示。当我们单击“应用”按钮时，所有选中的软件包都会被安装（连带它们的依赖性包）。

单击“应用”安装这个软件包之后，图 7-10 所示的确认对话框就会出现。

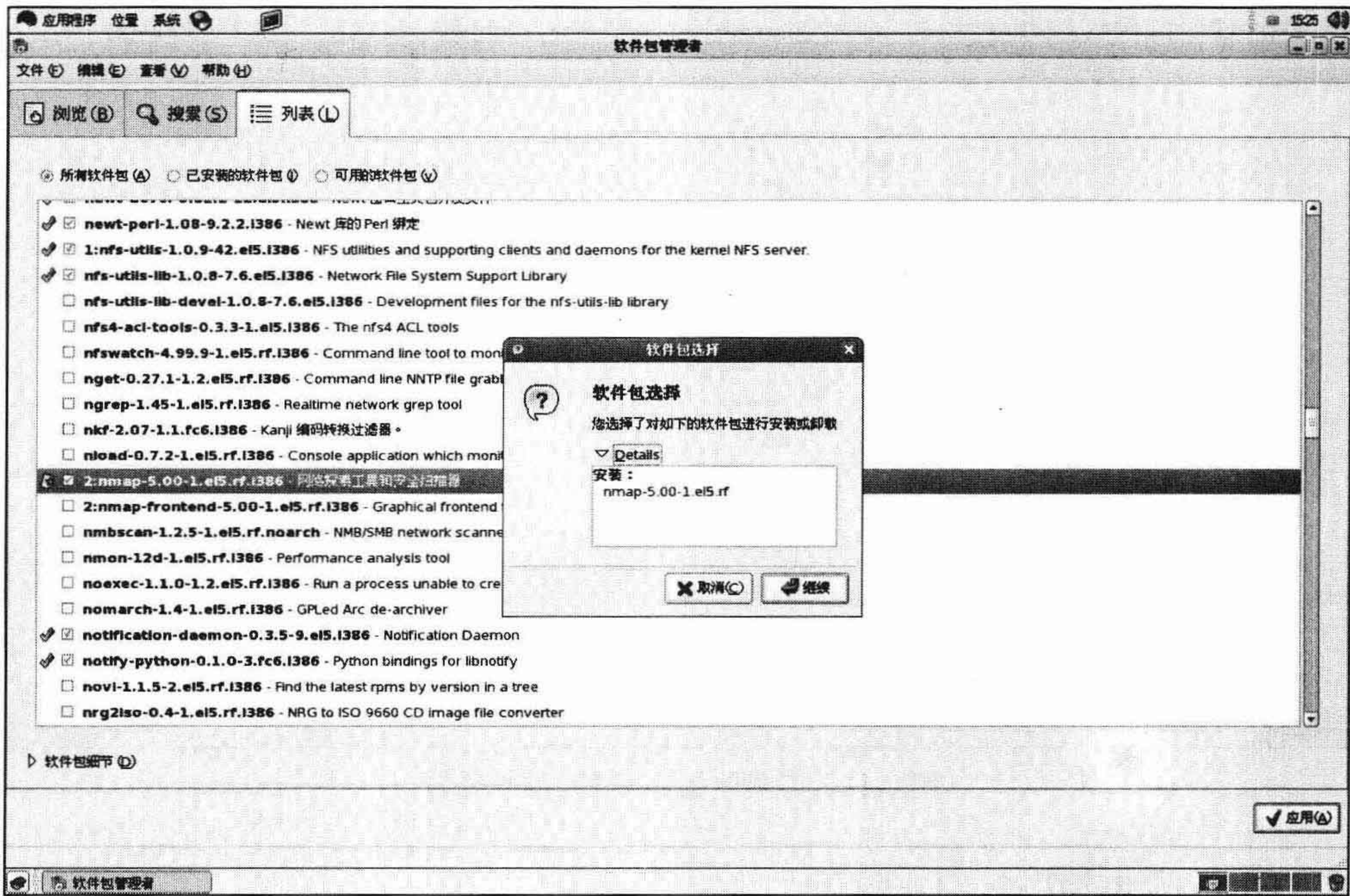


图 7-10 确认软件包管理器里的安装选择

要使用软件包管理器移除一个应用程序，可直接使用“搜索”标签查找已安装软件包，选中“已安装的软件包”单选按钮并在搜索字段中输入要移除软件包的名字。你还可以使用“列表”标签移除软件包。

图 7-11 显示，我们从已安装软件包组中搜索了软件包 **nmap**，并通过取消选定此软件包旁边的选择框表示要移除它。单击“应用”将移除这个应用程序，然后我们会得到一个类似于图 7-10 所示的确认画面。

你还可以使用软件包管理器查看已安装软件包、列出可用软件包以及搜索主机上已安装的或可用的软件包。软件包管理器提供了一个管理主机上软件包的快速而有效的方式。

如果单击“浏览”标签，你可以发现它与第 2 章后面为安装进程第一次选择软件包时的那个对话框（见图 2-17）非常类似。只要有合法的订购并连接到 Internet，你就可以看到对主机可用的软件包列表以及对每个软件包的描述（否则，将看不到任何信息）。在这里你可以通过选中复选框来选择单独的软件包或软件包组（如 Gnome 桌面或 Office 生产力软件组），并单击“应用”安装或删除它们。

提示：可以使用命令行搜索软件包、更新与添加软件包或者从主机移除软件包。本章稍后的“Yellowdog Updater Modified (Yum)”一节将介绍这些内容。



图 7-11 用软件包管理器移除软件包

7.2.4 Red Hat 网络 (RHN)

如果你正在使用 Red Hat Enterprise Linux，那么就会希望配置 RHN。这个基于订购的服务提供 Red Hat 主机的定期更新与补丁程序。RHN 有一个稳定的发布周期，并提供与系统运行相关联的成本预算功能。它还提供 Red Hat 工程师的支持，如果你遇到困难了，它还提供登门拜访的帮助。要使用这个服务，主机必须能访问 Internet。

■注：如果你没有在用 RHEL，可以跳过这一节。

有了 RHN，你就可以通过 Red Hat 的门户网站 <https://rhn.rehat.com> 管理一个或者数千个系统。从这个门户网站你可以看到系统状态以及与安全补丁相关的状态，利用它还能下载可访问的软件。本节还将介绍如何从这个门户网站管理软件包。

在使用 RHN 之前，你必须把主机注册到 Red Hat。你可能在第 2 章第一次启动主机时已经完成了这一步，但这里再介绍一遍，以防万一。

正如第 2 章所述，你在购买 Red Hat 软件时会收到授权，每个授权可以注册一台主机以接收软件更新与补丁程序。

要注册系统（如果在第 2 章安装主机期间没有这么做的话），你需要启动本节前面介绍的软件更新器工具。它会建立一条到 RHN 的连接，并验证主机是否拥有有效的订购。如果有订购，它就会呈现软件包更新器程序。如果没有，它会呈现一连串的画面让你把主机注册到 RHN。

如图 7-12 所示，第一个画面是一个欢迎界面，它提供了有关软件更新注册的简要信息。

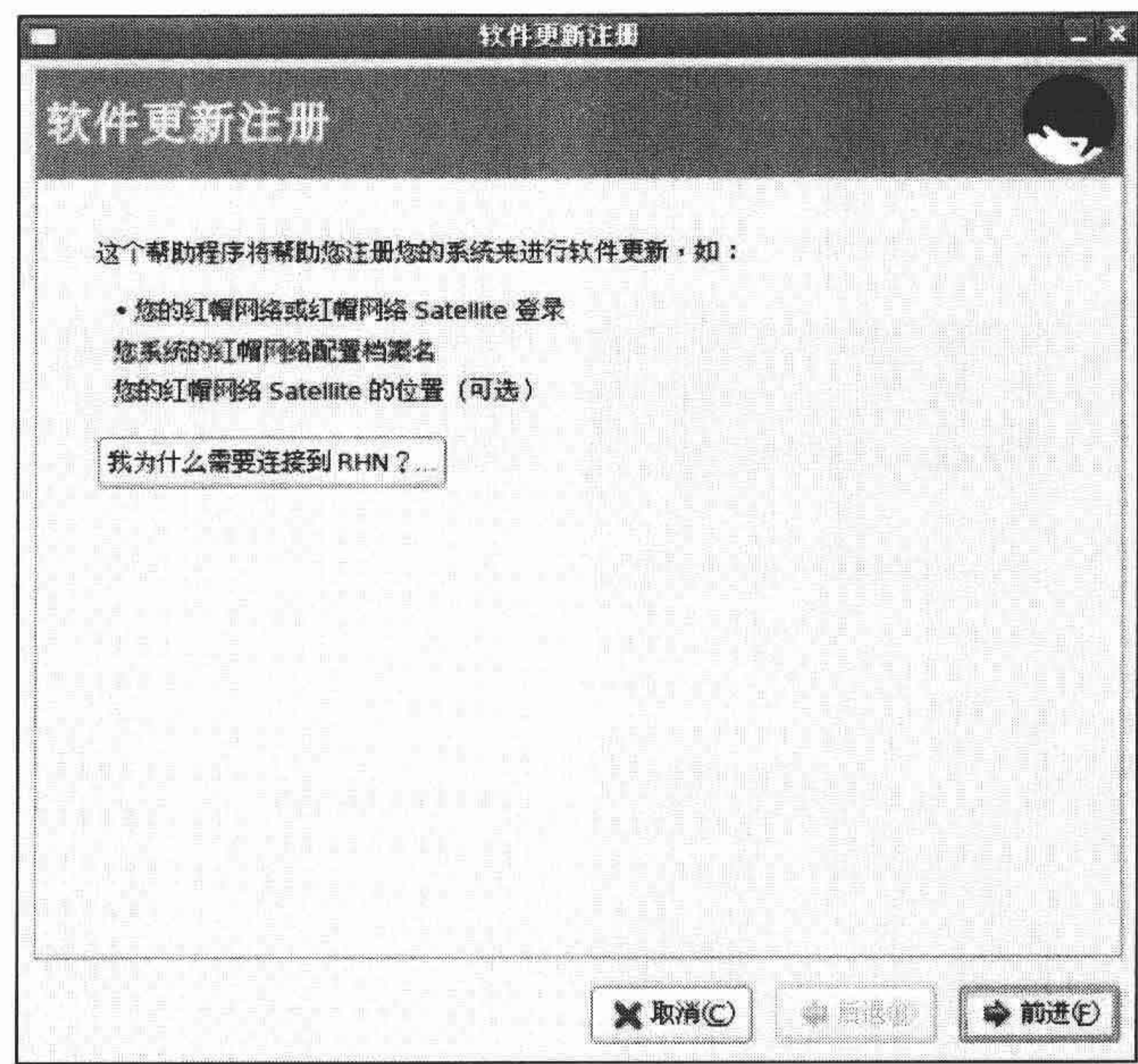


图 7-12 软件更新注册的第一个窗口

在接下来的窗口选择更新源的位置。在这里你可以选择 Red Hat 网络，或者使用一个 RHN 卫星服务器或 RHN 代理服务器。这最后两个是 Red Hat 应用程序预备组件的一部分，你可在管理多台主机时使用。图 7-13 中选择了 RHN。

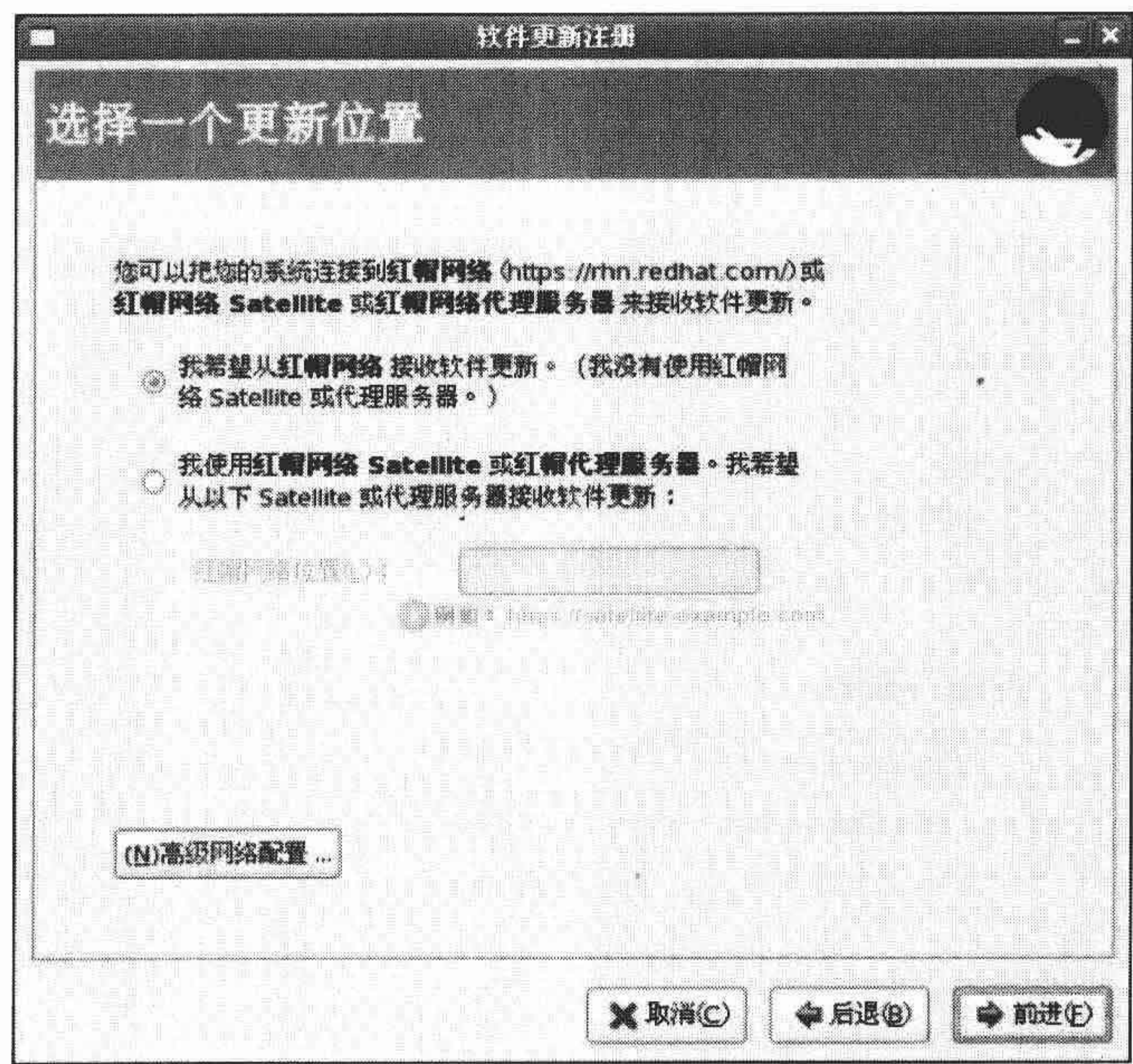


图 7-13 选择更新位置

现在它会提示输入用户名与密码，就是进入 RHN 门户网站所使用的那个用户名与密码。也许你以前已经注册了一个用户名与密码，也许正如第 2 章所描述的那样安装主机时已经创

建了它，否则可以马上到 <https://rhn.redhat.com> 创建一个。

■注：只注册不能拥有更新的权利——还需要购买软件并给主机分配授权。Red Hat 站点有相关的操作说明。

图 7-14 中指定了用户名 username。在这个位置要放置 RHN 用户名。

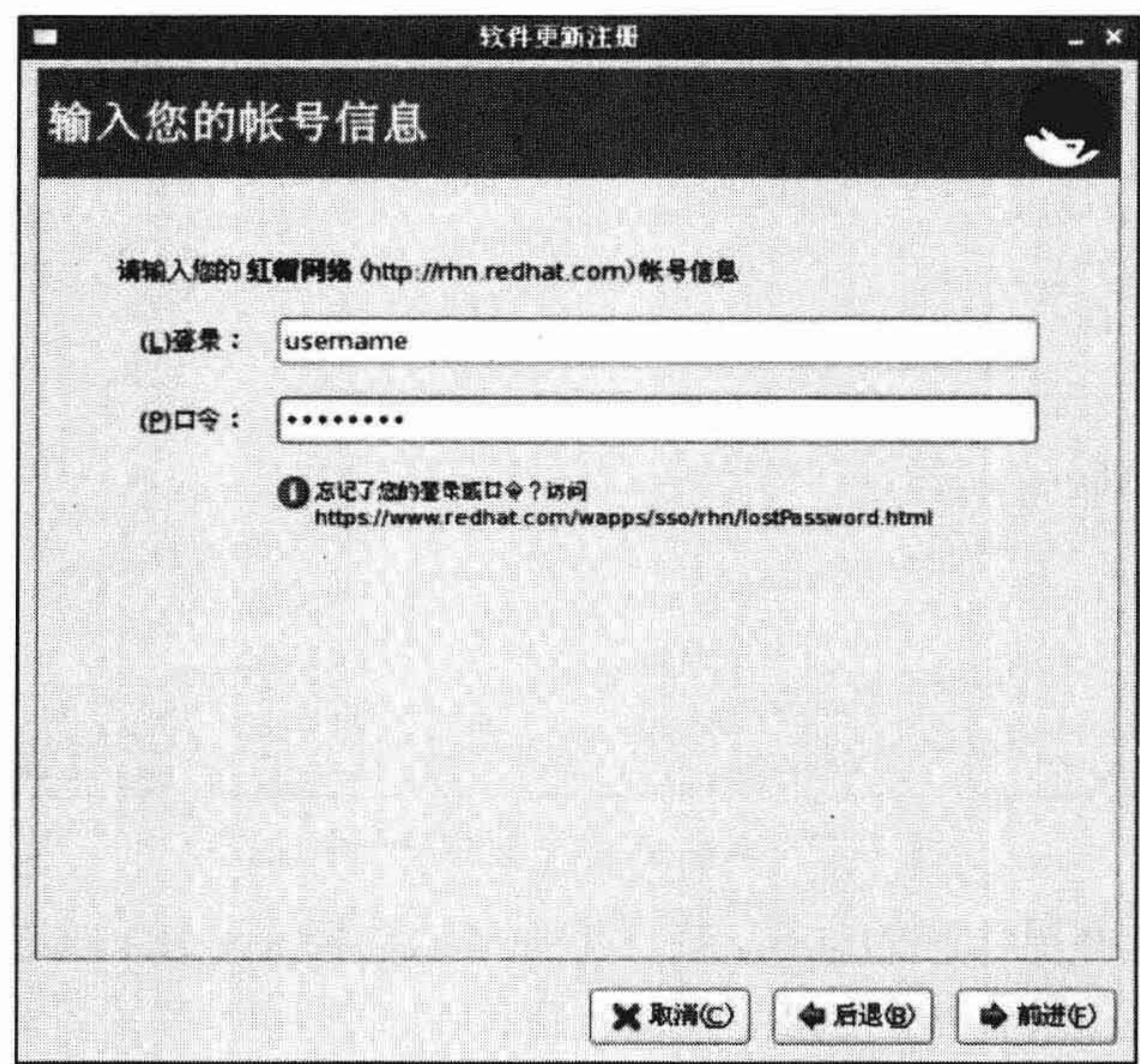


图 7-14 输入用户名与密码

接下来，你有机会输入系统名以及选择是否希望把软、硬件信息发送给 Red Hat。这个信息的用途是获得适用于系统的更新，推荐选中这些复选框，如图 7-15 所示。

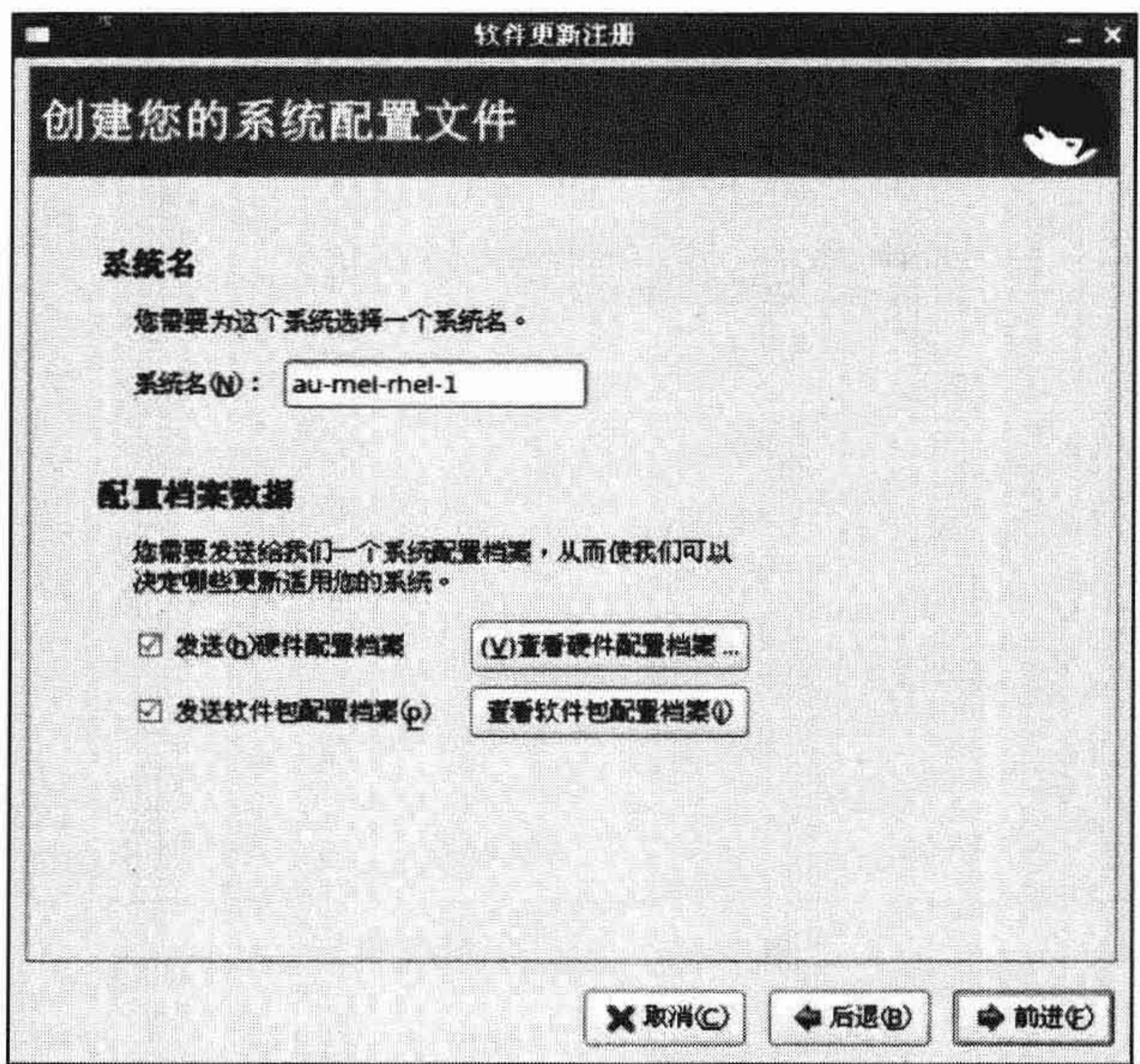


图 7-15 系统名与软、硬件信息

如图 7-16 所示，它提供了一个复查订购信息的机会。这很可能与你自己的订购信息不同。

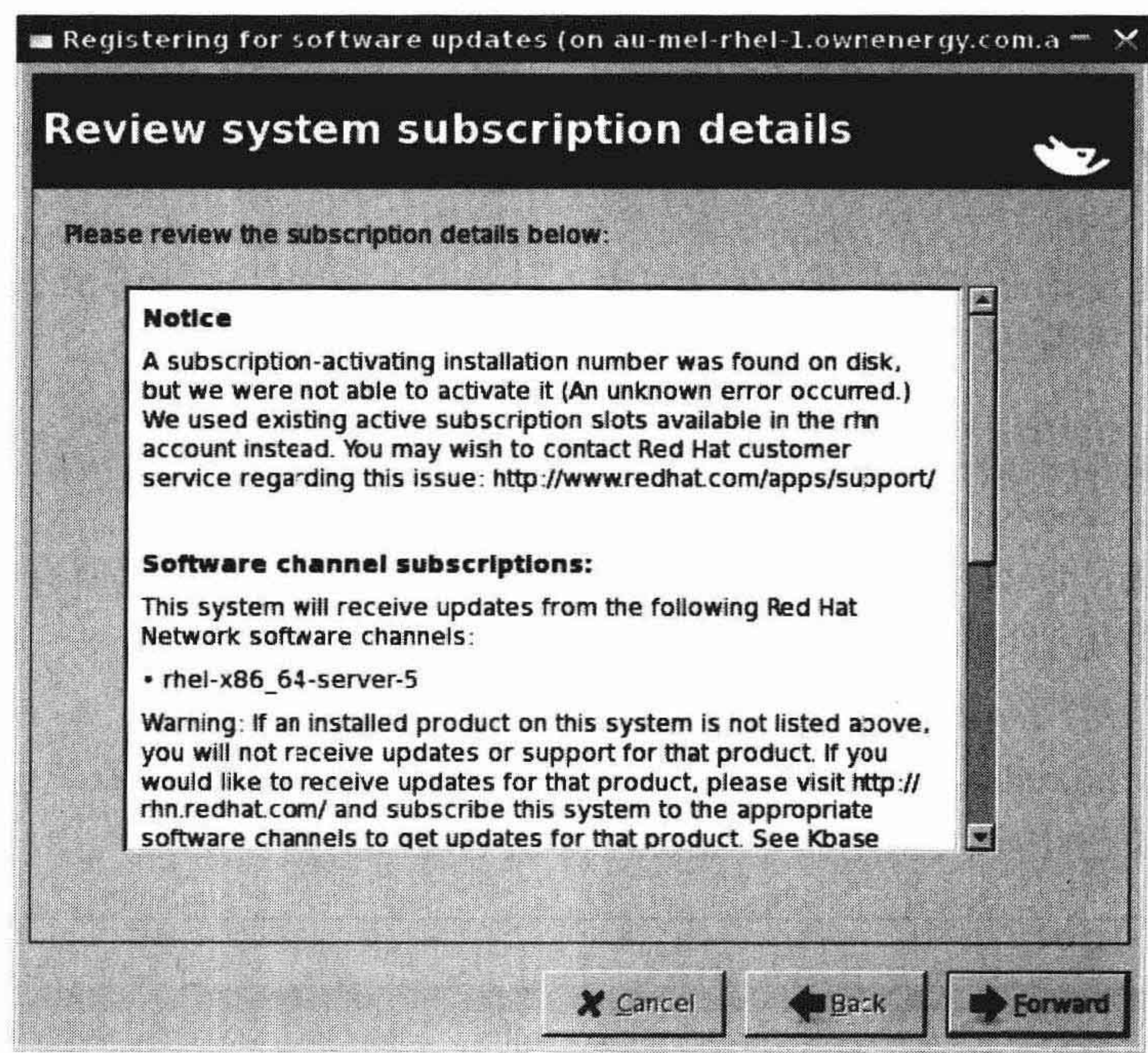


图 7-16 复查订购信息

现在它显示已成功设置系统更新的确认信息，如图 7-17 所示。如果你收到任何有关激活不成功的错误消息，应联系 Red Hat 的维护部门。

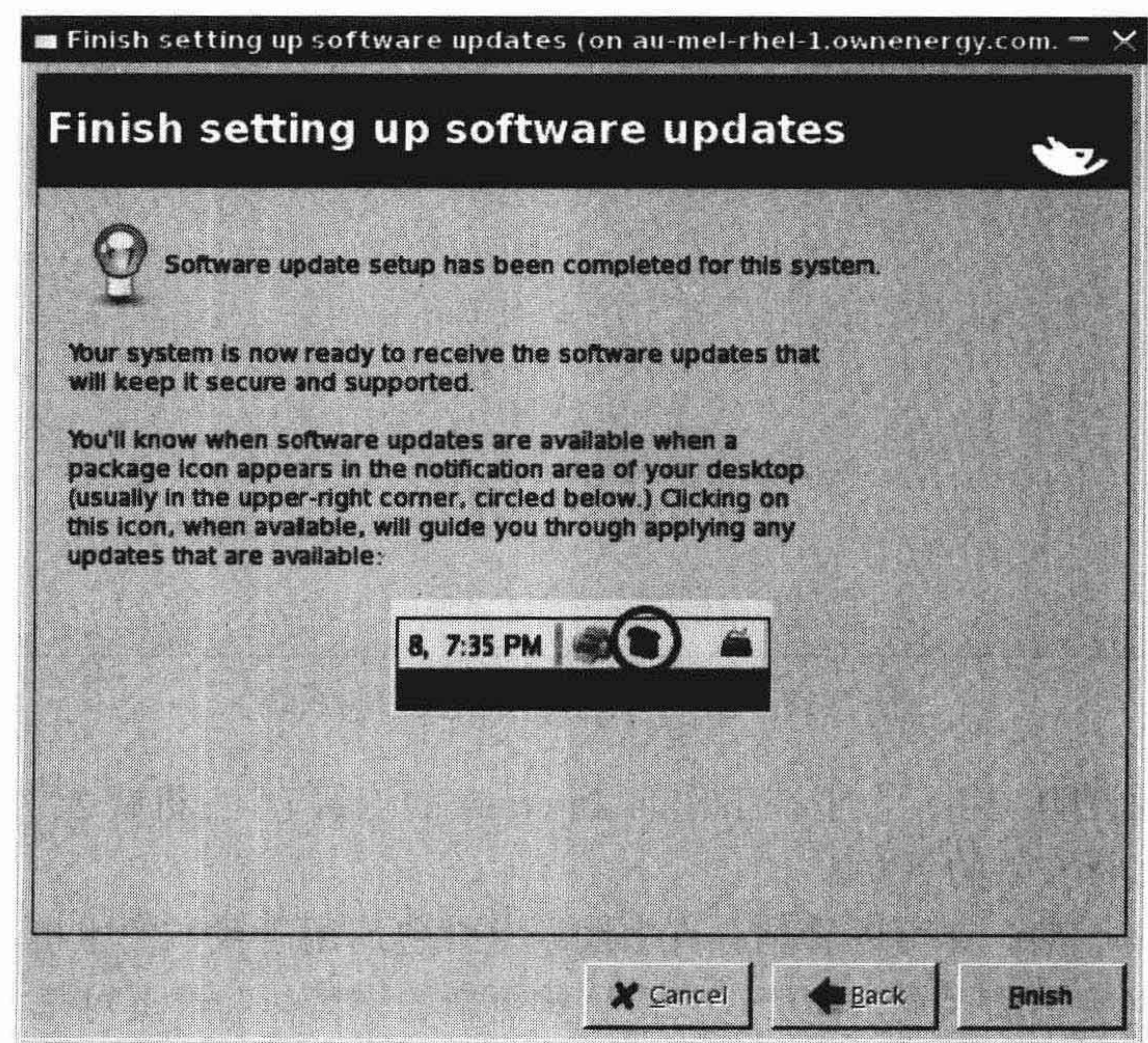


图 7-17 祝贺画面

补充知识：通过命令行注册

你可选择使用命令行替代 GUI 的方法注册系统。这是一个使用 `rhnmeg_ks` 命令快速达到同样结果的例子。首先你可以通过给 `rhnmeg_ks` 命令传递`--help` 开关查看可用选项。

```
$ sudo rhnmeg_ks --help
```

它会显示所有的选项。我们感兴趣的主要选项是 `username`、`password`、`profilename` 和 `activationkey`。通过使用这些选项，你可以看到从命令行注册系统是多么容易。在这里要把尖括号`<>`内的选项都替换为你自己的资料。

```
$ sudo rhnmeg_ks --username=<username> --password=<password> ➡
--profilename=<au-mel-rhel-1> --email=<sysadmin@yourdomain> ➡
--activationkey=<keyfromRHN>
```

一旦注册完成，你就可以使用 `https://rhn.redhat.com` 门户网站管理所有的软件维护权利。你可以查看哪些系统拥有哪些软件包及其当前版本。主机可以自动登入、下载重要的安全补丁，并安排从该门户网站进行软件包的安装与删除。

当登入 `http://rhn.redhat.com` 并定位到 `Systems` 时，你可以看到新系统已经注册。图 7-18 显示 `au-mel-rhel-1` 已注册并且有 37 个勘误包和 51 个软件包可用。

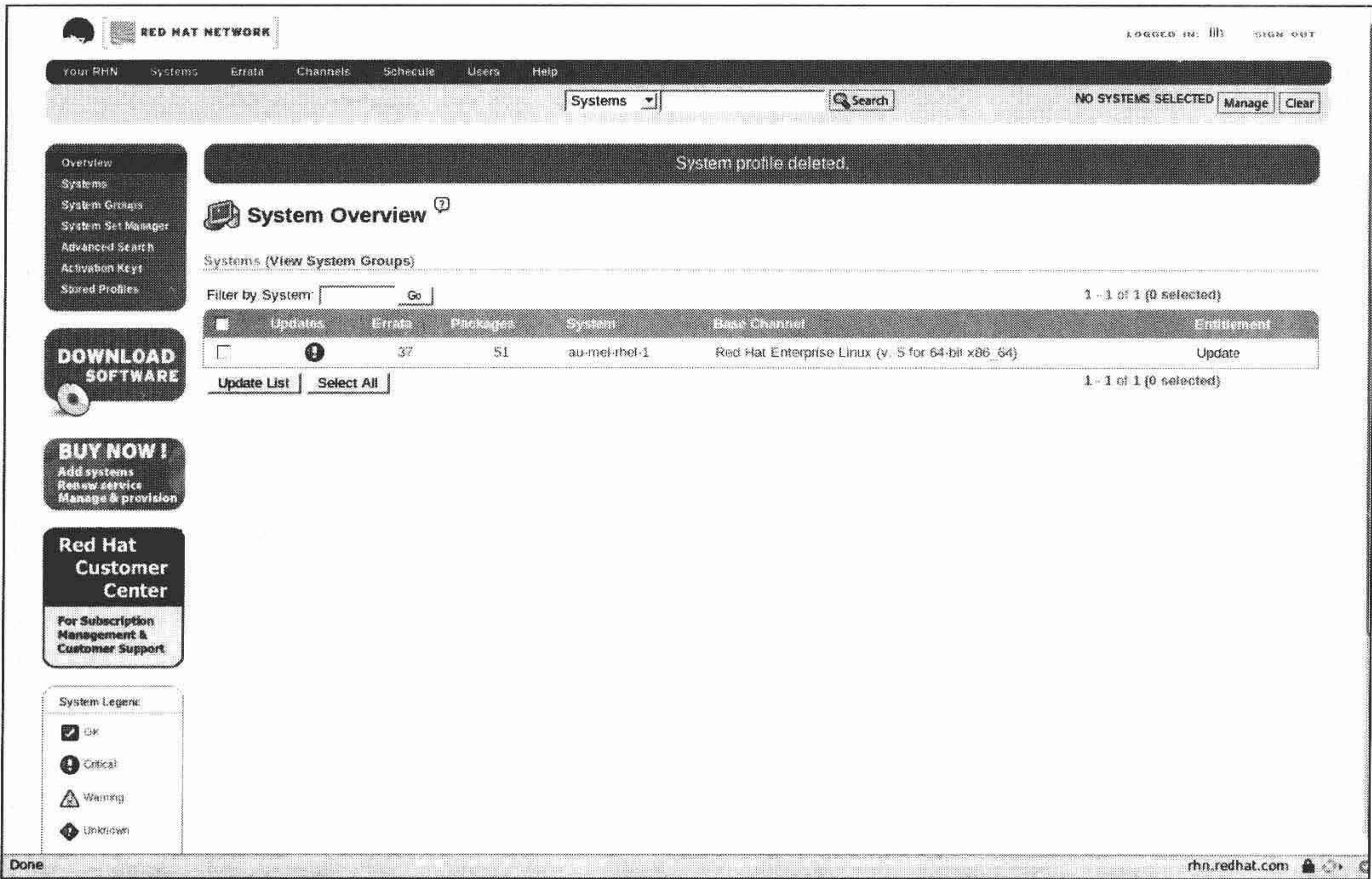


图 7-18 RHN 现在记录了我们的系统

只要有有效的订购（和一个正在工作的 Internet 的连接），主机就会立刻自动查询 RHN 并确定是否有任何待安装的更新。

从图 7-18 中的信息，你可以看到有许多勘误软件包适用于我们的主机。通过单击勘误栏的序号（如 37）链接到勘误页，你可以获得这些更新的详情。勘误页对每个更新都给出一个简短概述，并提供一个更详细的报告链接（见图 7-19）。

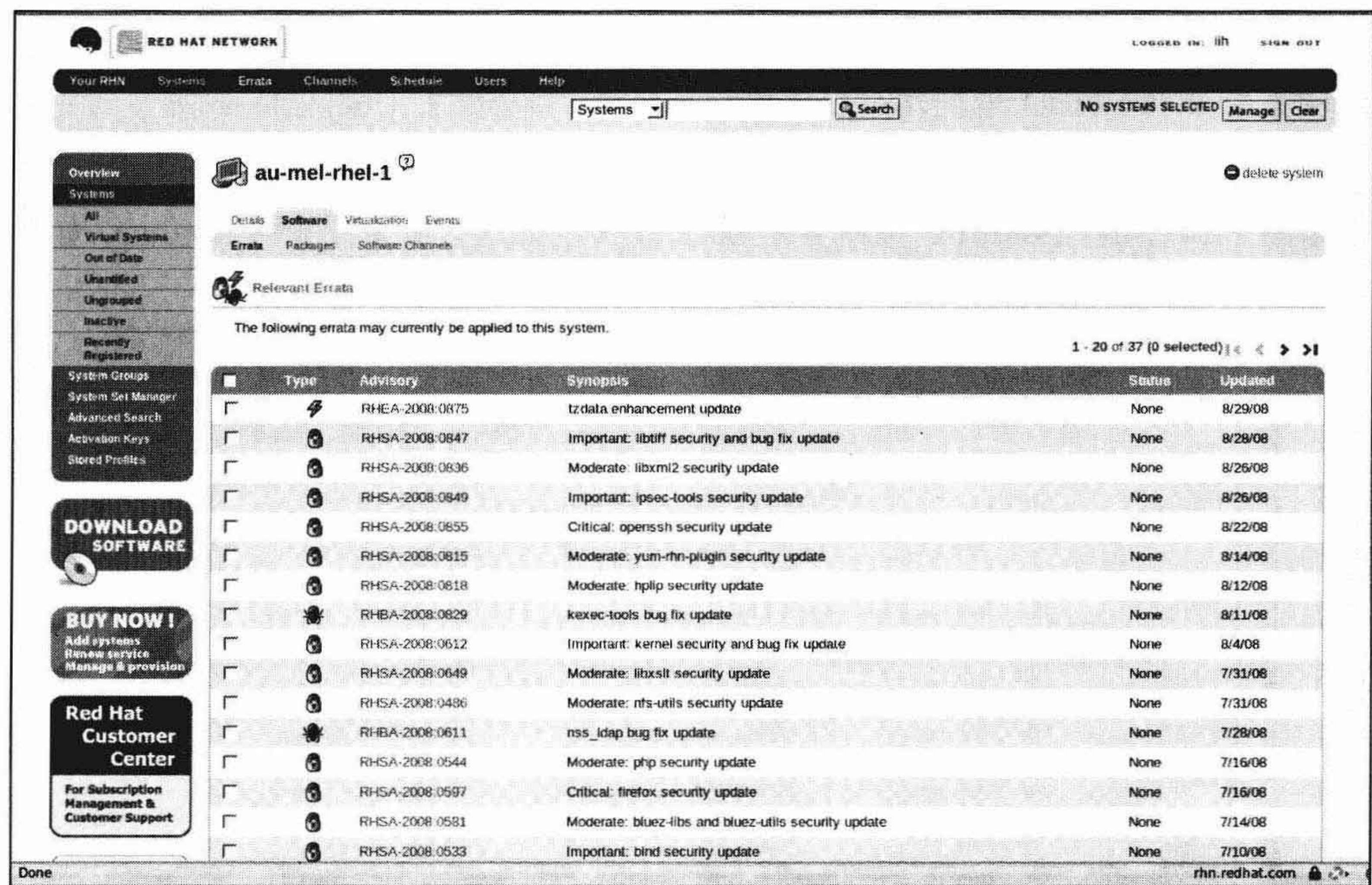


图 7-19 系统的 RHN 勘误页

再回到图 7-18 中的 RHN 页，注意红色感叹号图标。它表明哪些主机需要更新。当主机周期性地核查 RHN 查看是否有任何新的更新时，它还会核查要执行的预定事件（更新）。你可以使用 RHN 门户安排主机更新这些勘误软件包，它们会在下次核查时执行。如果单击图 7-18 所示的感叹号图标，它会转到 Confirm Errata Updates 页。在这个页面的底部你可以看到一个“Confirm”按钮，如图 7-20 所示。

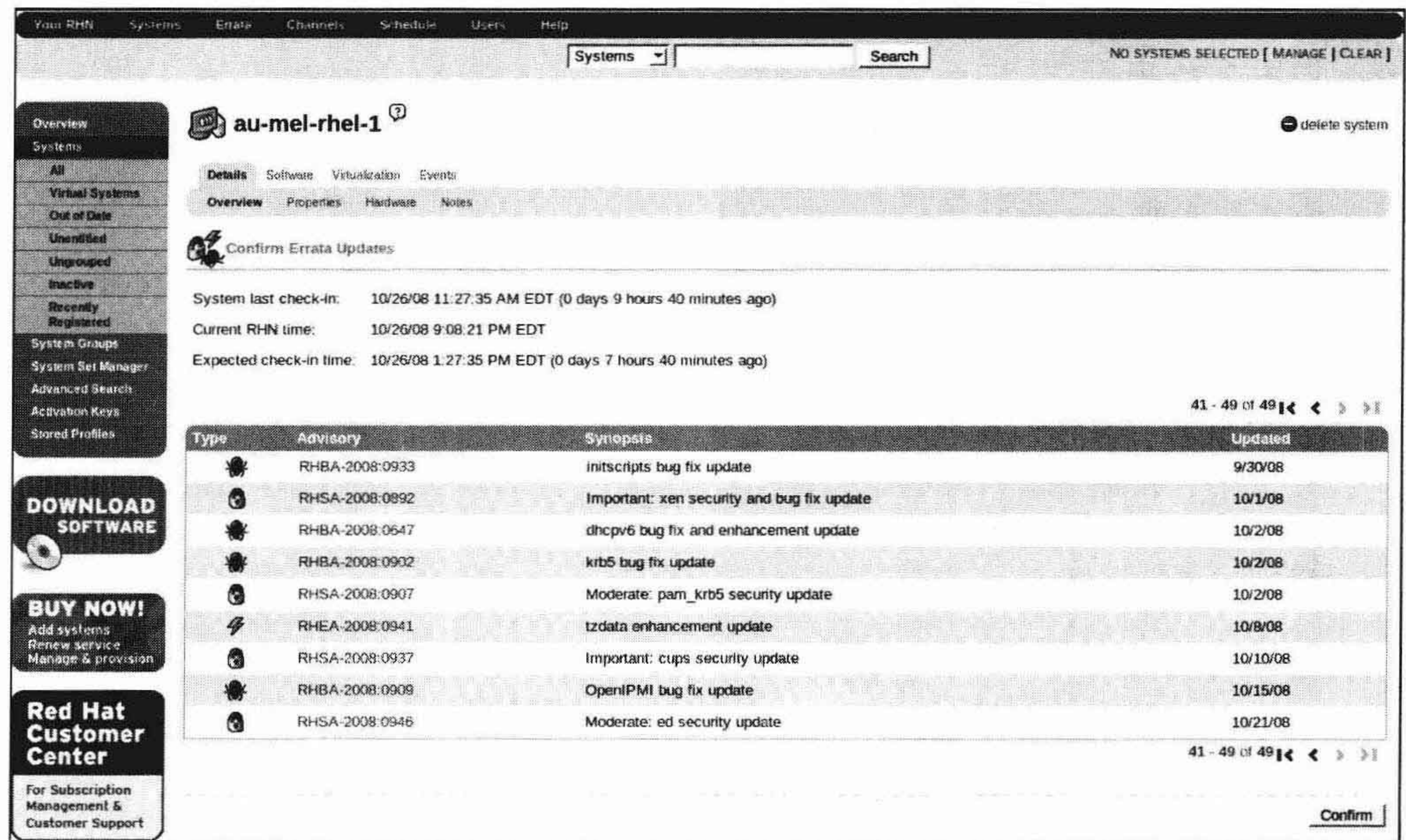


图 7-20 确认要预定的更新

■注：本节所概述的这个过程不是安排更新使之下次核查时执行的唯一途径，你可以在许多与主机相关的 RHN 页面上看到这个红色感叹号图标。那些配有 Update Now 文字的更新会自动确认并把这些事件添加到时间表。主机上所有未决的事件可以在系统的 Events 页面看到。

单击图 7-20 中的“Confirm”按钮会在 Scheduler 里把这些更新设置为 Pending。如图 7-21 所示，System Software Errata 页面上的这些更新现在处于 Pending 状态。

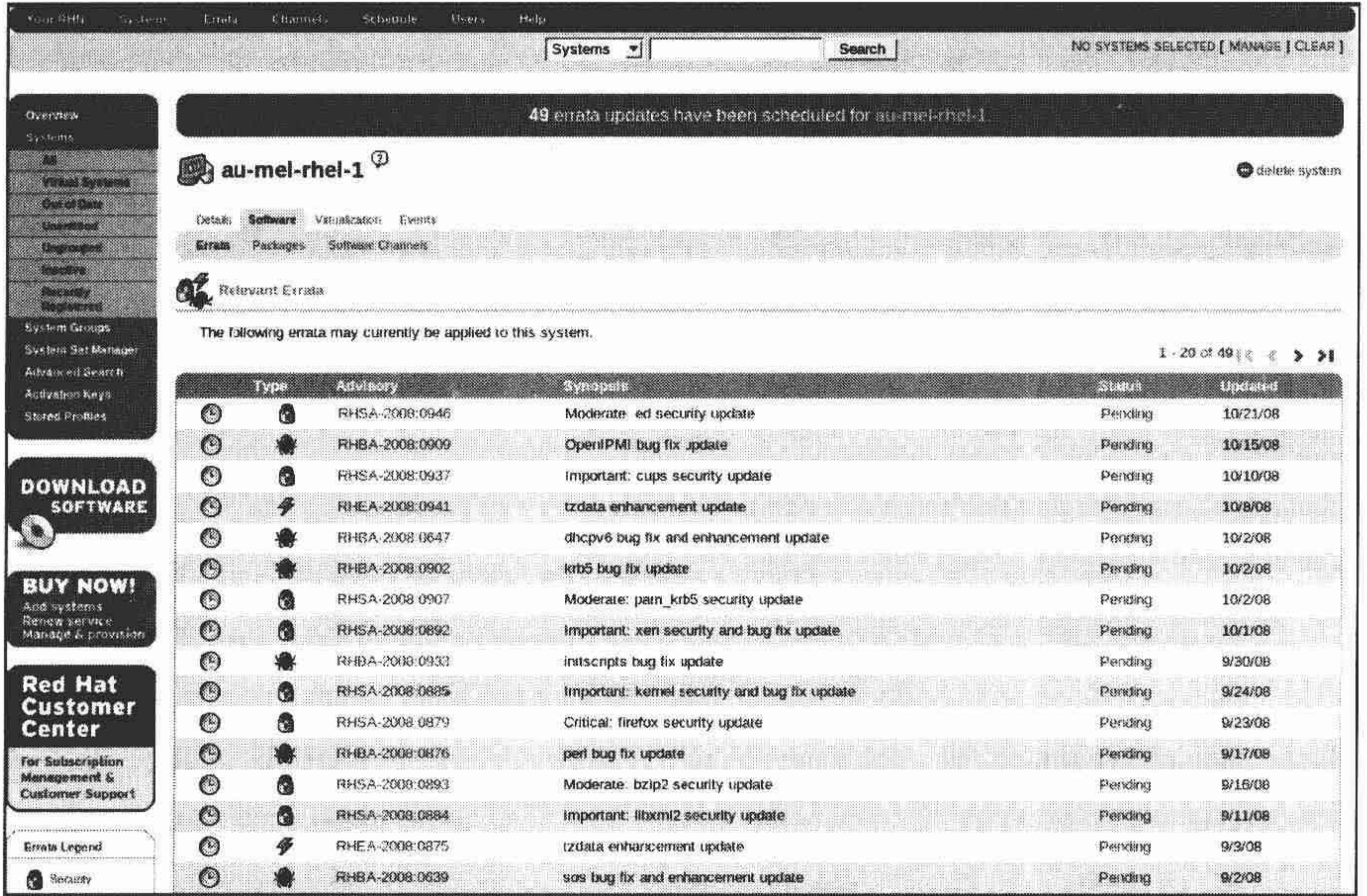


图 7-21 未决的勘误更新

你还可以到 System Events Pending 页面选择希望删除的所有更新或个别更新，取消这些更新事件。通过单击“Select All”按钮，然后单击 Cancel Events 按钮，我们取消了所有未决的事件，如图 7-22 所示。

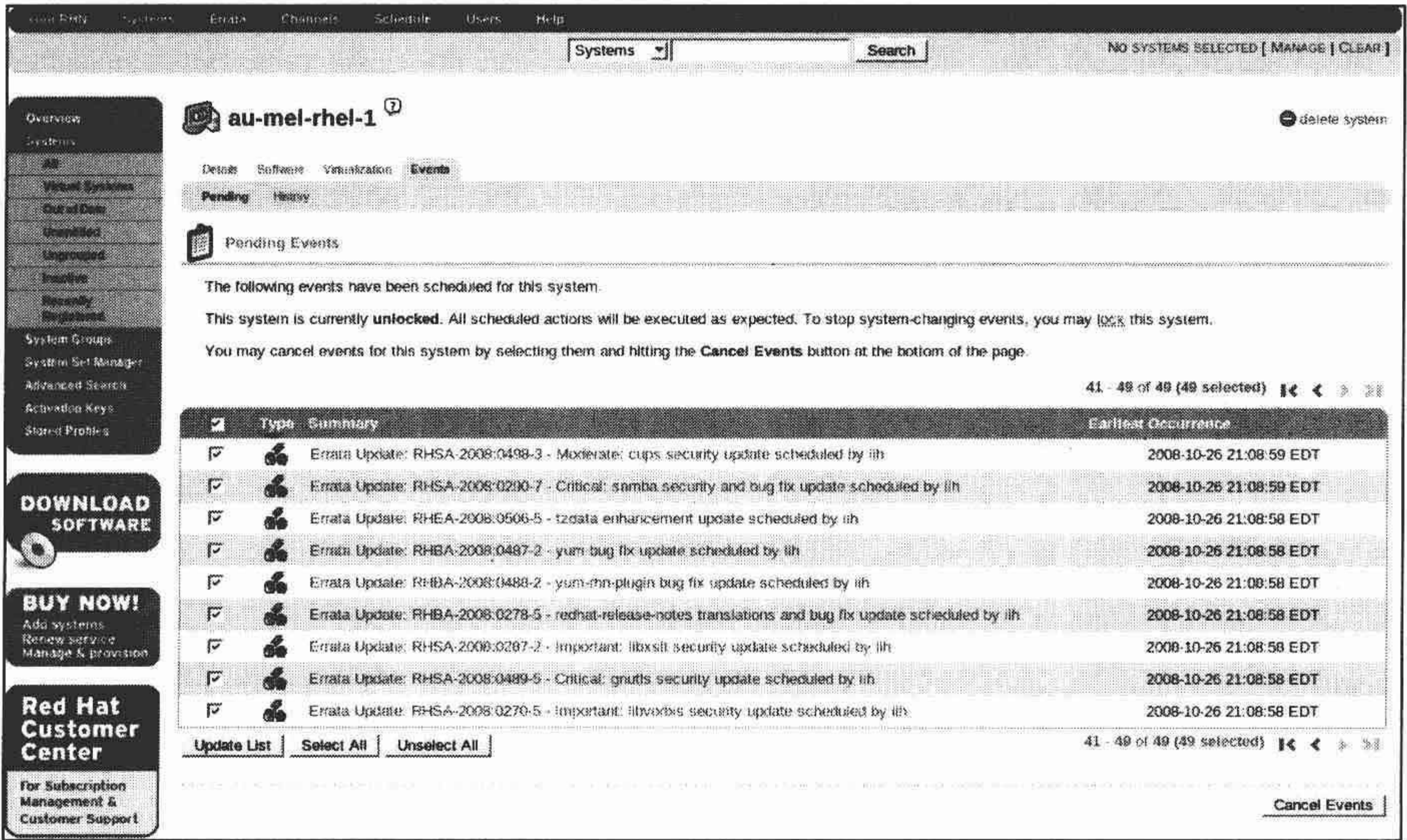


图 7-22 选择要取消的事件

单击“Cancel Events”按钮之后，你就会来到确认页面，那里显示已选择的更新和可以使用的“Cancel Selected Events”按钮。通过单击“Cancel Selected Events”按钮，你就确认了下次核查时不需要这些适用于主机的更新。

如果想再预定这些更新，你可以到 System Software Errata 页面再次选择并确认它们。事实上，你可以到 System Software Packages 页面从 Install New Packages 页面上的列表中选择希望安装的软件包，然后安排这些要安装到主机上的新软件包。对于我们的主机，我们有几乎 2000 个软件包可供安装。

要安装一个新软件包，你首先用英文字母索引定位到所希望安装的软件包。例如，你想安装 nmap，它是一个易用的网络映射工具。如图 7-23 所示，你可以单击所需的软件包，然后单击“Install Selected Packages”。它会进入一个确认页面，在那里一旦确认了，一个新事件就会被添加到时间表。

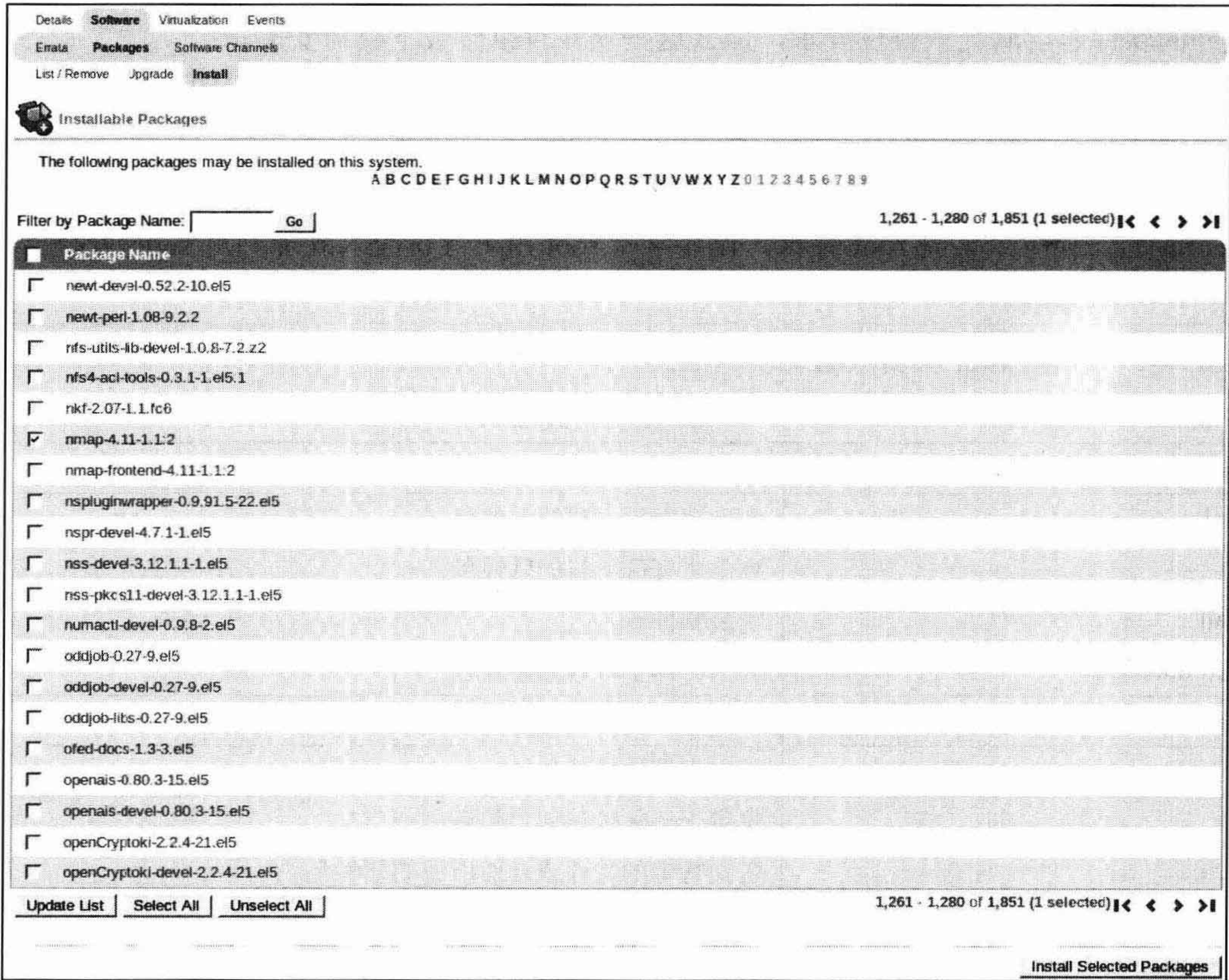


图 7-23 通过 RHN 门户安装新软件包

现在你可能开始看到通过 RHN 门户管理主机的妙处。你甚至没有登入主机就更新了它并添加了新软件。按照这样的方式使用 RHN 门户，考虑管理成千主机的那些可能性不会花费太长时间。通过对主机分组，你可以对组采取上述操作，那些更新或附加软件会自动安装到组内的全部主机。

7.2.5 Yellowdog Updater Modified (Yum)

对基于 Red Hat 的发行版，用来安装、升级或更新系统的最常见工具之一是叫作 Yellowdog Updater Modified (Yum) 的基于命令行的工具。Yum 提供在远程软件资料库上搜寻系统可用软件包的能力。如前所述，软件资料库就是由发行版厂商、兴趣组、大学和 ISP 制作的可用于发行版的软件包集合。

■注：Yum 可以在 Red Hat Enterprise Linux、Fedora、CentOS 以及其他 Red Hat 衍生的发行版上使用。它通常是这些发行版上通过命令行更新与管理软件包的默认方式。

Yum 还能够通过抓取安装任何个别软件包所需的软件包解决依赖性问题。它使用一个数据库来查询主机上已安装的软件包，然后拿它与软件资料库提供的软件包列表对比。如果需要额外的软件包以满足依赖性要求，那么这些软件包也会被下载与安装。

Yum 容易使用。在命令行上直接键入 yum 和一个可能的操作，如 install、remove 或 list，然后是希望对其采取动作的软件包的名字，如下所示。

```
# yum install nmap
```

■注：运行 yum 命令需要 root 特权，你可以以 root 用户身份登入或者利用 sudo 命令。推荐使用第 4 章演示过的 sudo 命令。

表 7-1 列出了 Yum 所提供的主要选项。

选项	说明
search	允许搜索可下载的软件包
list	列出所有可下载的软件包
install	安装软件包
check-update	列出可用于更新的软件包
update	更新指定的软件包或下载并安装任何可用更新
remove	从主机移除软件包
info	提供一个软件包的包信息
provides	告知特定的文件或功能所属的软件包
list updates	只列出有可用更新的所有软件包
list available	只列出所有可用软件包
localinstall	安装已下载到本地的 RPM 软件包
clean all	清理不再需要的已下载软件包文件

接下来，我们将介绍每个操作及其用法。

1. 用 Yum 安装软件包

让我们从尝试安装一个软件包开始使用 Yum。在正常情形下，我们首先会搜索软件包以

确保它可用。我们已经选择安装 `nmap` 软件包（如前所述，它是一个有用的网络映射与扫描工具）。我们使用 `search` 选项看它是否可以使用，如列表 7-1 所示。

列表 7-1 用 Yum 搜索软件包

```
$ sudo yum search nmap
```

■注：在列表 7-1 中没有使用 `root` 用户，而是使用第 4 章所演示的 `sudo` 命令来运行 `yum`。

在列表 7-1 中，我们发出了带 `search` 选项的 `yum` 命令，然后指定希望搜索的软件包名。它会产生一个名字中含有 `nmap` 的可用软件包列表。`yum` 命令没有智能到精确知道你正寻找的软件包，于是它提供了所有名字中包含 `nmap` 的软件包。

如果不确定软件包的完整名字，你还可以搜索部分名字。因此，举例来说，你可以键入 `yum search map` 以获得一个所有描述或名字里含有 `map` 字样的软件包列表。

■提示：你可以使用第 3 章所介绍的 `grep` 命令改进这个软件包列表。例如，你可以键入 `yum search php | grep pear`，它通过搜索 `pear` 这个词然后显示软件包列表结果，改进了与 PHP 相关的软件包列表。

你还可以使用 `yum list` 查看软件资料库中所有可用的软件包，而且还会发现 `yum grouplist` 对查看软件包组有用。软件包组是单独软件包的集合，这便于提供某种应用程序功能。那么举例来说，安装 KDE 组就可提供运行 KDE 桌面所需的一切，安装 Text-based Internet 组可提供用控制台访问网络应用程序（例如基于文本的浏览器 Lynx 和邮件客户端 Mutt）所需的常用软件包。

有了这个信息，我们要安装 `nmap` 软件包，就可以运行列表 7-2 里的命令。

列表 7-2 安装 nmap 软件包

```
$ sudo yum install nmap
$ sudo yum groupinstall 'Text-based Internet'
```

第一行会下载并安装一个叫作 `nmap` 的软件包，第二行会安装 Text-based Internet 组里的所有软件包。如果需要任何依赖性软件包，它们也会被下载，并且它会询问是否也想安装它们。如果没有安装所有必需的依赖性软件包，那么你就不能安装所需要的那个软件包。

2. 更新软件资料库

有时软件资料库的缓存数据会因为软件资料库里添加与更新了新软件包而过时。一个好的习惯是使用列表 7-3 中的命令刷新你的软件资料库，然后再执行软件包的安装。

列表 7-3 保持软件资料库更新

```
$ sudo yum check-update
```

一旦更新了软件资料库，你就可以运行下面的命令要求 Yum 安装任何可用更新。

```
$ sudo yum update
```

你还可以更新一个或多个单独的软件包，如列表 7-4 所示。

列表 7-4 用 Yum 更新软件包

```
$ sudo yum update nmap
$ sudo yum update nmap mutt
```

■注：你可以使用 Yum 把一个发行版的版本升级到另一个版本（例如从 Fedora 9 到 Fedora 10）。不过，我们不推荐这种方式，因为这有风险，可能导致对主机的损害。如果想进一步了解这种方式，请参阅 <http://fedoraproject.org/wiki/YumUpgradeFaq>。

3. 用 Yum 移除软件包

你还可以用 Yum 移除软件包，如列表 7-5 所示。

列表 7-5 用 Yum 移除软件包

```
$ sudo yum remove nmap
```

这个软件包被移除之后，列表 7-5 中所使用的 `remove` 选项会在主机上遗留几个文件，主要是配置文件。这意味着如果重新安装该软件包，你就不必重新创建配置。如果希望移除所有文件，你可以使用 `erase` 选项。

```
$ sudo yum erase nmap
```

4. 执行其他 Yum 任务

使用 Yum 命令还可以做许多其他的事情。让我们简要地浏览一部分命令。

下面的命令提供所查询的软件包信息，如它的大小、版本和安装状态，一个软件包说明，以及其他有用信息。

```
$ sudo yum info kernel
```

如果希望查明是什么软件包提供了某个特定文件，你可以使用下面的命令。

```
$ sudo yum provides /bin/bash
bash-3.2-22.fc9.x86_64 : The GNU Bourne Again shell (bash) version 3.2
Matched from:
Filename : /bin/bash
```

上述命令告诉你 `/bin/bash` 二进制文件是由 `bash` 软件包提供的，并说明了该软件包的版本与详情。

为了查看主机的可用更新列表，你要发出下面的命令。

```
$ sudo yum list updates
```

如果主机是最新的，这个命令将没有结果。

接下来，你可以发出下面的命令以列出可用来安装到主机上的所有软件包。

```
$ sudo yum list available
```

上述命令列出软件资料库中还没有为主机安装的所有可用软件包。

最后，你可以清理缓存目录（Yum 暂时存储已下载软件包的地方）。如果需要收回一些

磁盘空间，你可以清理这个缓存。

```
$ sudo yum clean all
```

上述命令移除/var/cache/yum 缓存目录里所包含的缓存软件包与头文件。

5. 配置 Yum

Yum 程序有多种配置方式。Yum 的配置文件存储在/etc/yum.conf 和/etc/yum.repos.d/中，状态文件存储在目录/var/lib/yum 和/var/cache/yum 中。

■注：状态文件告诉 Yum 哪些软件包已经安装到主机，另外它还保存已下载软件包的缓存版本。要小心不要删除这些目录，因为这很容易破坏你的软件包数据库。

通常你不需要修改默认的/etc/yum.conf 配置。你可能希望改变配置的一个例子是添加 Yum 下载软件包时要使用的代理服务器。默认设置一般适合大部分环境。通过 yum.conf 的 man 页你可以得到 Yum 可用配置设置的完整列表。

```
$ man yum.conf
```

尽管很少改变 Yum 的配置，但你可能希望给 Yum 添加别的软件资料库以便从中下载其他软件包。定义 Yum 可用软件资料库的文件包含于/etc/yum.repo.d 目录。让我们看一个通用软件资料库文件。

```
$ cat /etc/yum.repo.d/myrepo.repo
[myrepo]
name=myrepo
baseurl=http://myrepo.mydomain.com/pub/linux/releases/$releasever/$basearch/os/
enabled=1
gpgcheck=1
gpgkey=http://myrepo.mydomain.com/linux/RPM-GPG-KEY-linux
```

对每个选项的解释见表 7-2。

表 7-2 添加 Yum 软件资料库的选项

选项	说明
[repo-id]	repo_id 是软件资料库的唯一名字
name	name 是对软件资料库的描述
baseurl	baseurl 是软件资料库的位置（URL）。它一般是一个 HTTP URL，十分类似网络浏览器中所使用的网址
enabled	启用或停用软件包，0 代表停用，1 代表启用
gpgcheck	这个选项告诉 Yum 检验用于软件包签名的 GPG 密钥，以确保软件包不被篡改。指定 0 关闭校验，指定 1 打开校验
gpgkey	这是 Yum 寻找软件资料库的 GPG 密钥的 URL

在列表 7-6 中，我们定义了一个新的 CentOS 发行版的软件资料库。

列表 7-6 在 Yum Repo 文件里添加一个 CentOS 软件资料库

```
[source]
name=CentOS-releasever - Sources
```



```
baseurl=http://mirror.centos.org/centos/$releasever/os/SRPMS/
gpgcheck=1
enabled=1
gpgkey=http://mirror.centos.org/centos/RPM-GPG-KEY-CentOS-5
```

你可以看到在 `baseurl` 选项中指定了一个叫作 `$releasever` 的变量。在 Red Hat 与 CentOS 上，此变量默认为 `redhat-release` 软件包的版本。在 Fedora 上，此变量默认为 `fedora-release` 软件包的值。在这两种情况中，此变量都表明了发行版的当前版本。例如，对于 Fedora 9，其版本如列表 7-7 所示。

列表 7-7 显示发行版的当前发布版本

```
$ sudo yum info fedora-release
Installed Packages
Name           : fedora-release
Arch           : noarch
Version        : 9
Release        : 5.transition
Size           : 65 k
Repo           : installed
Summary        : Fedora release files
URL            : http://fedoraproject.org
License        : GPLv2
Description: Fedora release files such as yum configs and various /etc/ files ➡
that define the release.
```

这个变量是一个可指定更通用 URL 的快捷方式。Yum 使用这个 URL 寻找正确的软件资料库，并用发行版版本替换 `$releasever` 变量。例如，在 CentOS 发行版版本 5 上，下面的网址：

```
http://mirror.centos.org/fedora/$releasever/os/SRPMS/
```

变成这个 URL：

```
http://mirror.centos.org/centos/5/os/SRPMS/
```

7.2.6 Red Hat 软件包管理 (RPM)

我们已经介绍了 `yum` 命令，也掌握了怎样使用它进行包管理。在很多情况下，`yum` 命令只是对另一个命令——`rpm`——的包装。Red Hat 中，`rpm` 命令是操作含有包的 RPM 文件的基本工具。虽然通常情况下你不会直接使用该命令，但是明白它的原理还是很重要的。

我们推荐读者先阅读一下 `rpm` 的 `man` 页，大体了解一下该命令。

```
$ man rpm
```

在 `rpm` 的 `man` 页中，你可以看到 `rpm` 工具所能完成的任务：查询、安装、升级和卸载。让我们仔细看看 `rpm` 命令的参数选项。表 7-3 显示了可以传递给 `rpm` 的主要选项以及它们的作用，每个选项都有一些标志符。

表 7-3 rpm 命令的主要选项和标志符

选项和标识符	描述
-q --query	允许用户查询 RPM 数据库，查找主机上软件包安装的信息
-i --install	本地安装软件包
-e --erase	卸载软件包
-U --upgrade	给已安装的软件包升级，或者如果软件包尚未安装则安装它
-F --freshen	如果已经安装了旧版的软件包，则对它进行升级
-V --verify	核查已经安装的软件包

1. 查询软件包

首先我们尝试用-q 或者--query 选项查询已经安装的软件包。rpm 工具通过查询一个数据库中的数据来查找主机上已经安装了哪些软件包。当有软件包被安装或者卸载时，该数据库便被更新。

■注：RPM 数据库只记录系统的当前状态；它并不记录昨天卸载了什么或者软件包之前的版本是什么等信息。这些安装、卸载、升级记录和关于之前的版本信息都记录在主机的日志文件中。你可以在/var/log/rpmpkgs 文件中查看 rpm 命令的动作记录。

我们现在用 rpm 命令检查 Linux 操作系统的内核模块——kernel。如果被问到系统安装了 Linux 哪一版的内核，你可以用 rpm 命令找到答案，如列表 7-8 所示。

■注：你也可以用 uname -r 命令查找正在使用的 Linux 内核的详细信息。

列表 7-8 查询内核版本

```
# rpm --query kernel
kernel-2.6.18-92.el5
kernel-2.6.18-95.el5
```

这里我们用--query 选项运行 rpm 命令，并且指定了想要查询的软件包名称——本例中的包名是 kernel。你可以看到该命令列出了一个已安装的内核列表。

■注：因为安装了多个内核，所以在列表 7-8 中，你可以看到不止一个内核。这并不代表同时运行多个内核，只是有多个可运行内核而已。

列表中每个内核末端的数字是其版本号。其中一个是 2.6.18-2，另一个是 2.6.18-95。这两个版本的内核都能被 Red Hat Enterprise Linux 系统 5 (.el5) 所用。

■注：第 5 章讨论如何选择想要启动的内核。

如果不知道所查找的软件包的包名，或者想要查看系统都安装了哪些软件包，你可以使用列表 7-9 的命令。

列表 7-9 查询所有软件包

```
# rpm --query --all
```

这条命令用了--query 和--all 选项，表明查询和列出所有已安装的软件包。

正如所见，该命令的查询结果列表可能会很长。假设只想找出那些包名中含有 `php` 的软件包，你可以把该命令的输出通过管道重定向到 `grep` 命令，然后用 `grep` 在该命令的输出中搜索字符串 `php`，如列表 7-10 所示。

列表 7-10 用管道和 `grep` 查询所有软件包

```
# rpm --query --all | grep php
php-common-5.1.6-20.el5
php-cli-5.1.6-20.el5
php-5.1.6-20.el5
php-ldap-5.1.6-20.el5
```

■注：第 3 章覆盖 `grep` 命令和管道的内容。

列表 1-10 显示了把 `rpm` 查询的输出通过管道重定向到 `grep` 命令，然后让 `grep` 查询字符串 `php`。这样把列表从几千个软件包减少到 4 个包名含有字符串 `php` 的软件包（列表为空代表在已安装的软件包中没有包名含有给定字符串的软件包）。

下面我们找出更多关于内核软件包的信息。在列表 7-11 中，我们把 `--query` 选项和 `--info` 选项结合，查找其中一个内核的更多信息。

列表 7-11 获取软件包信息

```
# rpm -q --info kernel-2.6.18-92.el5
Name           : kernel                Relocations: (not relocatable)
Version        : 2.6.18                Vendor: Red Hat, Inc.
Release        : 92.el5                Build Date: Wed 30 Apr 2008 04:10:41 AM EST
Install Date: Fri 15 Aug 2008 11:18:29 AM EST
Build Host     : ls20-bc2-13.build.redhat.com
Group          : System Environment/Kernel Source RPM: kernel-2.6.18-92.el5.src.rpm
Size           : 82381879              License: GPLv2
Signature      : DSA/SHA1, Thu 01 May 2008 07:42:12 AM EST, Key ID 5326810137017186
Packager       : Red Hat, Inc. <http://bugzilla.redhat.com/bugzilla>
URL            : http://www.kernel.org/
Summary        : The Linux kernel (the core of the Linux operating system)
Description    :
The kernel package contains the Linux kernel (vmlinuz), the core of any Linux operating system. The kernel handles the basic functions of the operating system: memory allocation, process allocation, device input and output, etc.
```

列表 7-11 产生了大量信息，尽管其中一些对我们而言可能没有多大意义，但还是来看看这些信息。你可以看到软件包（本例中是 `Red Hat`）的版本号、生厂商，以及它被创建和安装的日期。你还可以看到该软件包隶属于哪个组，因为每个软件包都要隶属于由相似软件包构成的组。第 2 章安装系统时，在选择安装软件包的这一步骤中，你可以看到这些信息。

而且，你可以看到软件包在哪个许可证下发行（本例是 `GPLv2` 许可）和软件包的大小，最重要的是软件包的描述和总结，以及一些关于软件包更多详情的链接。

有时候想知道一个特定的文件或者命令由哪个软件包安装，你同样可以用 `rpm` 工具来查询这一信息。比如，如果安装了 `/bin/bash` 命令，那么我们可以用 `rpm` 命令找出安装该命令的

软件包是什么，如列表 7-12 所示。

列表 7-12 使用 query 和 whatprovides 选项

```
# rpm --query --whatprovides /bin/bash
bash-3.2-21.el5
```

列表 7-12 告诉我们文件/bin/bash 是由软件包 bash 提供的，同时它还显示了 bash 软件包的版本号。

现在知道 bash 软件包提供了文件/bin/bash，然而系统中还有哪些文件是由 bash 软件包提供的呢？有了--whatprovides 所提供的信息，我们就可以通过使用--query--list 选项查看哪些文件属于 bash 软件包，如列表 7-13 所示。

列表 7-13 使用 query 和 list 选项

```
# rpm --query --list bash
/bin/bash
/bin/sh
/etc/skel/.bash_logout
<snip> ...
```

列表 7-13 列出了一个省略的文件列表，这些文件都是由 bash 软件包提供的。

2. 安装软件包

rpm 工具也可以用来安装和升级软件。安装软件包时，你需要下载所需要的 RPM 文件，然后用 rpm 工具安装它。

■注：如果使用 yum 命令或者能添加、删除软件包的 GUI 工具，那么相应的 RPM 文件会在安装时自动下载完成。这就是用这些工具比直接用 rpm 命令要好的原因之一。

为了示范，我们用 wget 命令下载一个 RPM 文件，然后安装它。

■注：wget 命令是个网络下载器（其表现像个 Web 浏览器），它可以获取网络站点的文件，但没有浏览器的其他功能。

在列表 7-14 中，我们从 Nmap 网站下载文件，然后列出下载目录中的文件，确认文件已下载成功。我们要下载两个版本的 Nmap 应用程序，这样就能首先展示如何用 rpm 安装应用程序，然后展示如何把它升级到一个新的版本。

列表 7-14 从因特网上下载软件包

```
$ wget http://mirror.centos.org/centos/5/os/SRPMS/nmap-4.11-1.1.src.rpm
```

列表 7-15 用命令 rpm --install--verbose--hash 安装 RPM 文件。

列表 7-15 用 rpm 安装软件包

```
# rpm --install --verbose --hash nmap-4.11-1.1.src.rpm
Preparing... ##### [100%]
1:nmap ##### [100%]
```


--install 选项用来安装软件包，--verbose 选项用来显示 RPM 命令所做的事，--bash 选项用来产生井号提示符（#）——提示安装进度。

■注：如果没有安装 Nmap 所依赖的其他软件包，那么该安装过程将会失败，并给出一个错误信息。这是用 Yum 或者 GUI 工具管理软件包要比 rpm 更好的另一个原因——Yum 和 GUI 工具能为我们处理安装和卸载所依赖的软件包。

3. 卸载软件包

我们还可以用 rpm 命令卸载软件包，如列表 7-16 所示。

列表 7-16 使用 rpm 命令卸载软件包

```
# rpm --erase nmap
```

--erase 选项用来把 nmap 软件包从系统中彻底地删除。表 7-4 给出了在卸载软件包时可能用到的其他一些选项。

表 7-4 配合--erase 使用的额外选项

选项	描述
--noscripts	禁用与该软件包相关联的所有脚本
--nodeps	安装之前不检查依赖项
--test	必须连同--vv 一起使用；在 rpm 运行过程中显示将要删除什么

表 7-4 中显示的第一个选项是--noscripts，它告诉 RPM 直接卸载软件包，而不要运行任何与该软件包相关联的卸载脚本。这样做可能会在系统中遗留一些文件。--nodeps 选项只移除该软件包而保留它所有依赖的软件包。最后，--test 选项允许尝试卸载软件包。如果同--vv 选项结合使用，它就会展示在 rpm 命令运行时将会删除哪些东西，而不会真正地卸载该软件包。

■注：本节对 rpm 工具做了非常简短的介绍。然而，我们并不推荐使用它；相反，推荐使用 GUI 工具和本章前面介绍的 yum 命令。

7.2.7 通过源代码创建 RPM 软件包

为什么要用源代码安装软件包？因为有时候我们需要通过源码给软件打补丁，或者想要编译软件包的最新版本。如果属于上述这些情况，就可以采用以下两种方法进行源码安装。

许多软件包含有称作预览版（upstream）的 RPM 文件。预览版 RPM 是那些含有应用程序最新版本的软件包——比发行版中的软件包要新。预览版 RPM 通常是由应用程序的开发者或者应用程序开发社区的其他成员创建。它们可能含有我们需要的新补丁或者新功能，但是没有发行版本稳定，也可能含有缺陷和问题。这些预览版 RPM 通常可以从应用程序的官方网站上获得。

■注：任何人都能创建预览版 RPM。所以这些文件可能不太稳定、不太安全，也得不到定期维护。使用它们的时候要小心。

第二种方法是下载源码并且通过源码创建自己的 RPM。这种软件包的创建超出了本书的范围，然而有些非常出色的网上资料可以帮助读者创建自己的 RPM 文件。

- Maximum RPM: <http://www.rpm.org/max-rpm-snapshot/>。
- RPM Guide: <http://docs.fedoraproject.org/drafts/rpm-guide-en/>。
- Packaging Software with RPM, Part 1: <http://www.ibm.com/developerworks/library/l-rpm1/>。
- Creating RPMs: http://pmc.ucsc.edu/~dmk/notes/RPMs/Creating_RPMs.html。

需要指出的是从源码创建自己的软件包需要耗费一定时间，这取决于软件包的复杂程度和它所含的依赖项个数。我们不阻止读者去尝试从源码创建软件包，但是警告读者：创建自己的软件包这条路有时候可能变得漫长而困难重重。

7.3 Ubuntu 上的软件包管理

Ubuntu 服务器的软件包管理通常是通过命令行工具操作的，因为 Ubuntu 服务器默认没有安装 GUI 模块。本节首先介绍 Ubuntu 上用来管理软件包的命令行工具。之后，讨论一个图形化的软件包管理工具，Synaptic。只要 Ubuntu 系统安装了 GUI 模块就可以使用 Synaptic 攻击。

Ubuntu 主机上安装软件最常见的方法是通过命令行工具从在线软件源安装软件。Ubuntu 的运营商提供了超过 22000 个软件包的在线软件源。多年来，出现了几个用来帮助用户从远程软件源安装软件的工具。这些工具提供了和 Red Hat 系统上的 Yum 相当的功能。下面将介绍两个命令行工具。

- aptitude
- dpkg

这两个工具在 Ubuntu 主机上是默认安装的。aptitude 工具能利用在线软件源进行安装、卸载、更新和查找软件包。它是管理软件包的首选工具，所以下面会详细讲解该工具。

dpkg 是用来安装和卸载软件包的最基本的命令行工具。aptitude 内部利用了 dpkg 来管理软件包。因为不会经常用到 dpkg，所以下面只简短介绍一下该工具。

aptitude

你可以用两种方法操作 aptitude 工具：通过菜单和对话框交互式地操作或者通过命令行给它传递指令。对于初学者，最简单的方法就是用 aptitude 基于菜单的用户接口。让我们启动 aptitude 看一看。

```
$ aptitude
```

在 aptitude 处理完软件包列表后，就会显示 aptitude 的主窗口，如图 7-24 所示。

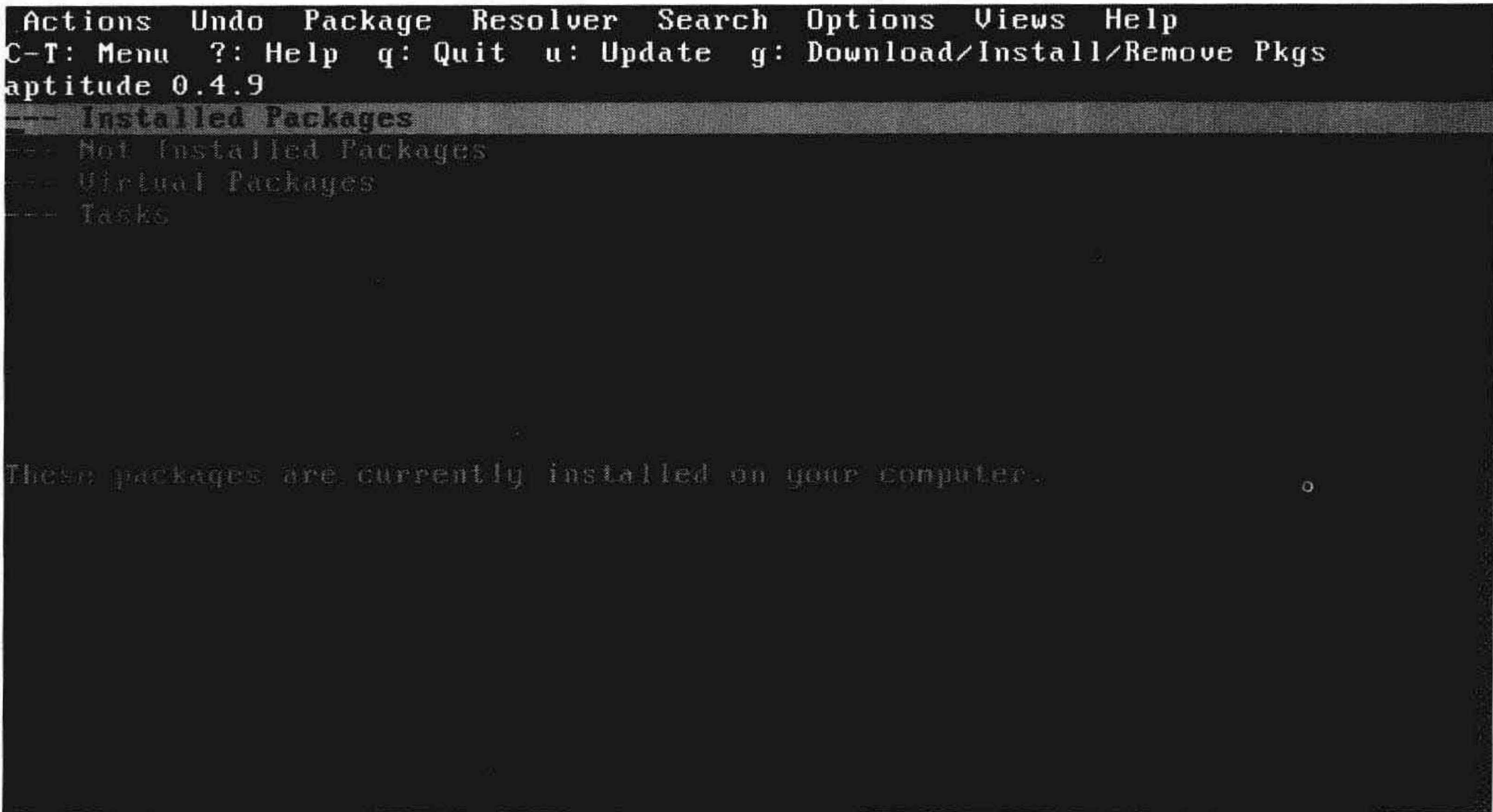


图 7-24 aptitude 的主窗口

该主窗口包含一个菜单栏、一个命令列表、一个窗口列表、一个状态信息区、一个软件包列表窗口以及一个信息窗口。光标将首先出现在软件包列表窗口。你可以用方向键上下移动选择列表项并按回车键展开当前高亮选中的列表项。

按回车键展开“Installed Packages”列表项，接着选中并展开“base”列表项，然后展开“main”列表项。将会在“main”下面看到当前已安装的软件包的一个按字母排序的列表。如果高亮选中一个软件包，该软件包的描述信息将被显示在信息窗口。按 Tab 键切换到该窗口，然后用方向键上下浏览信息。再次按 Tab 切换到列表窗口。在图 7-25 中，你可以看到高亮选中了“adduser”软件包。

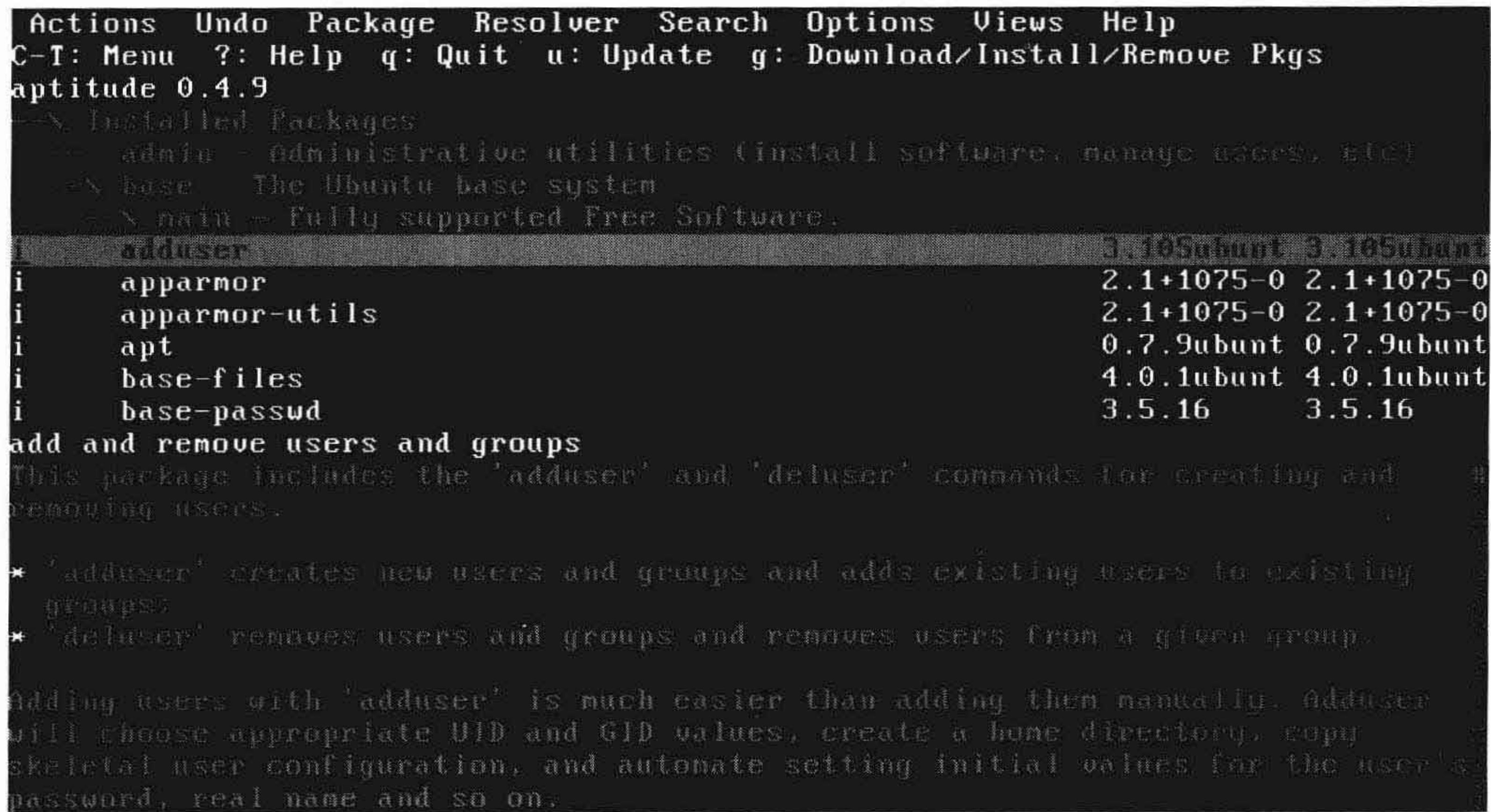


图 7-25 显示软件包描述信息

为了显示软件包更多的详细信息，高亮选中包名并且按回车键，如图 7-26 所示。你可以看到依赖项、维护人员等一些信息。

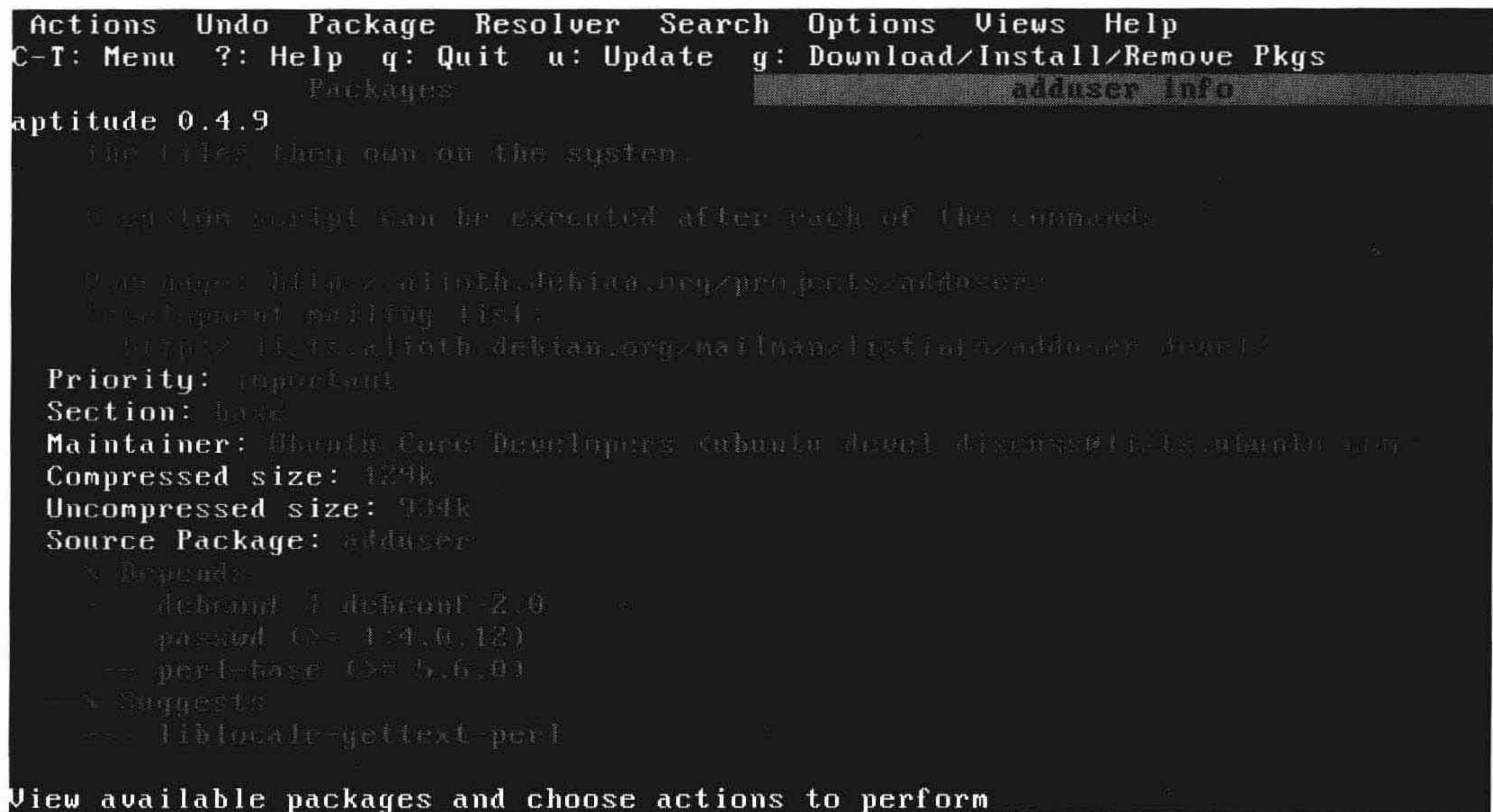


图 7-26 显示软件包详细信息

按 q 键关闭信息窗口并返回软件包列表窗口。

由于滚动浏览该列表不是很快，支持软件包的搜索就有了实际的用处。通过按正斜线 (/) 开启搜索对话框并输入所需的搜索词。现在查找 Ubuntu 的内核模块软件包 linux-image。输入搜索词并按回车键，aptitude 就会把列表下移，显示第一个匹配的软件包。可以看下图 7-27 的例子。

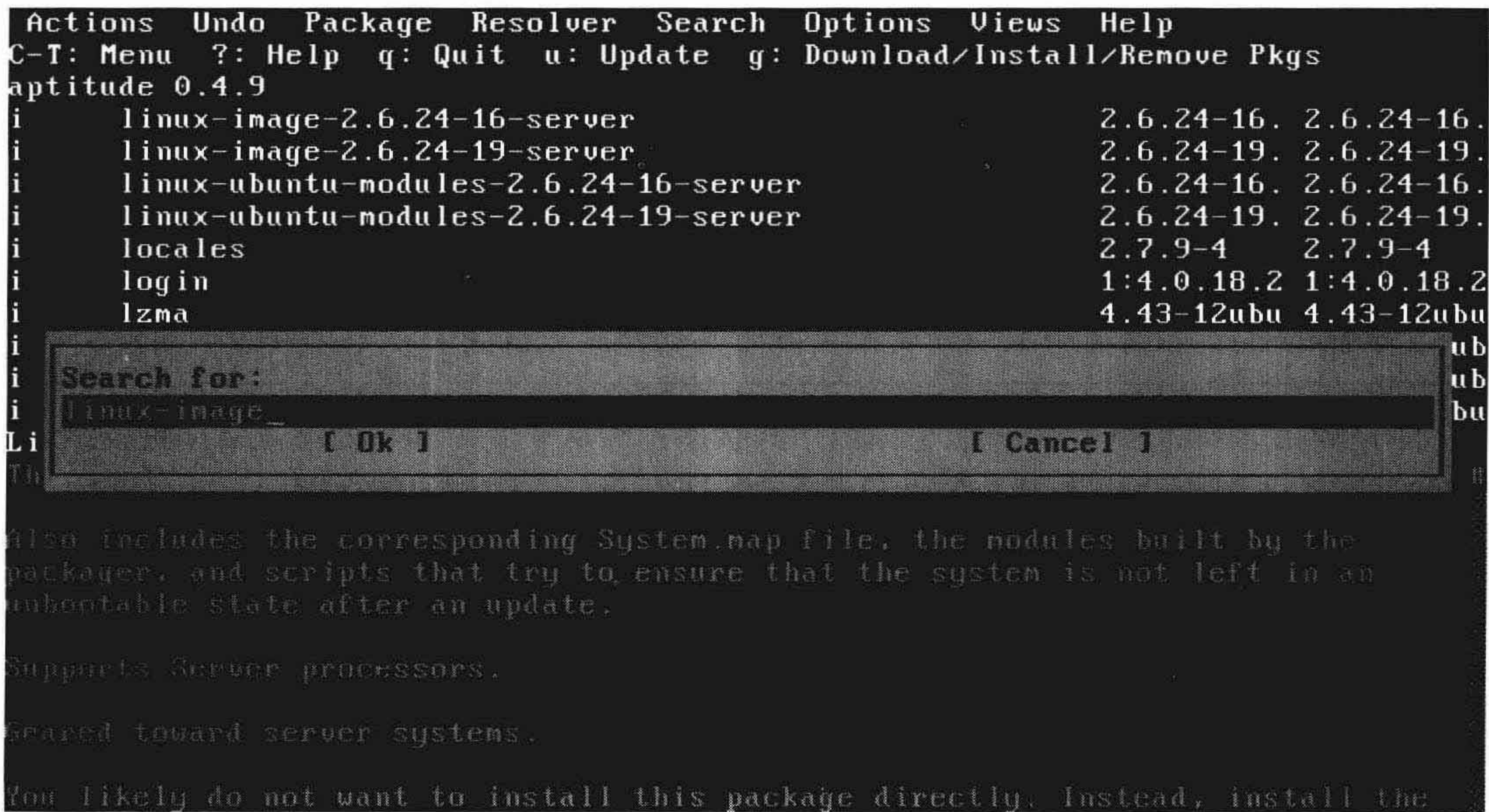


图 7-27 通过名称搜索软件包

按回车键关闭查询对话框。按下 `n`，转到下一个匹配的软件包。如果一直按 `n`，最后会看到未安装的软件包列表。该列表包含更多的内核软件包。为了在同一屏幕中显示更多软件包名称，你可以按下 `D` 关闭信息窗口。

现在我们试着安装一个软件。我们选择安装一款小巧实用的网络工具，`nmap`。使用 `find` 命令在列表中找到该软件包，然后高亮选中它。按下加号（`+`）标记要安装该软件包。你可以看到一个警告对话框，它告诉我们只有 `root` 用户才可以用 `aptitude` 安装软件，如图 7-28 所示。

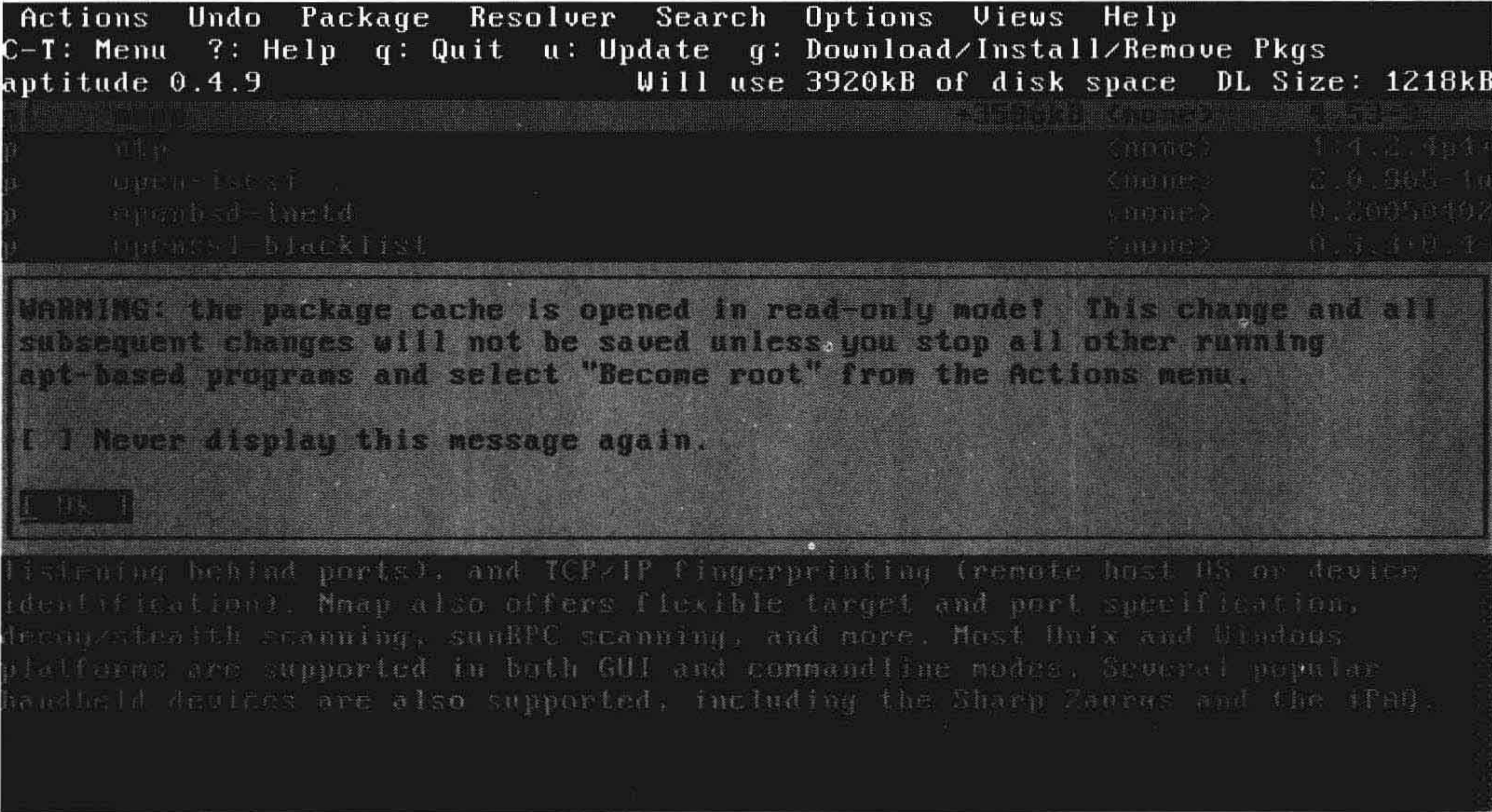


图 7-28 警告对话框

你可以禁用该警告，让它以后不再出现。注意到这里有个“Never display this message again”复选框。首先按 `Tab` 键转到该复选框，按空格键选中它，然后再按 `Tab` 键选中“OK”按钮并按回车键启用设置。你也可以按回车键去了解该警告。

注：如果通过 `sudo` 运行 `aptitude`，该警告对话框就不会出现。然而，以非 `root` 用户操作则可以添加一层防止出错的保护层。因为如果不是 `root` 用户就不太可能意外卸载软件，以非 `root` 用户运行 `aptitude`、在需要使未决的更改生效时再切换到 `root` 用户是个不错的做法。

你可以看到该软件包已经被做了标记，其状态已经从 `p` 变为 `pi`，而不是立即被安装。这允许我们在对系统所做的更改生效之前选择安装或卸载任意数量的软件包。

现在就安装 `nmap` 这个应用程序，首先你必须切换到 `root` 用户。为了切换到 `root` 用户，按 `Ctrl+T` 组合键激活动作菜单。用上下键选择菜单项“Become root”，并按回车键，`aptitude` 将会运行 `sudo` 命令。输入 `sudo` 密码之后，`aptitude` 将以 `root` 用户权限重新启动。虽然 `aptitude` 的软件列表返回到了最初的状态，但是之前安装软件的命令已经被添加到了 `aptitude` 内部的 `to-do` 列表中。

按 `g` 继续安装。`aptitude` 会发现 `nmap` 软件包依赖 `libpcrc3`，因此 `libpcrc3` 被自动选择（被标记为 `iA`），如图 7-29 所示。

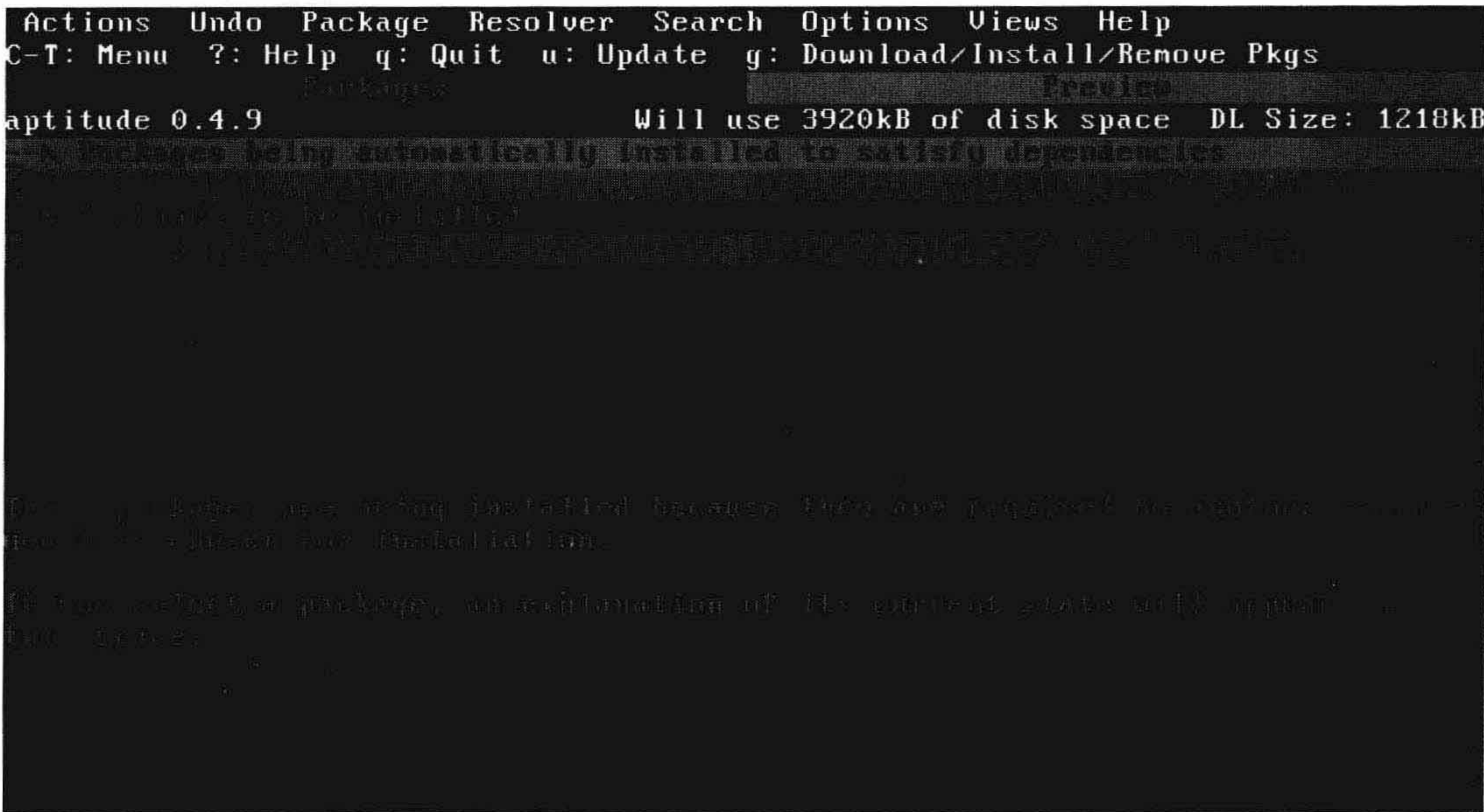


图 7-29 添加依赖软件包

再次按 **g**，确认需要安装该依赖项。现在 **aptitude** 会下载所需的软件包文件然后安装它们。在安装过程中，我们会被告知安装过程所发生的事（见图 7-30）。

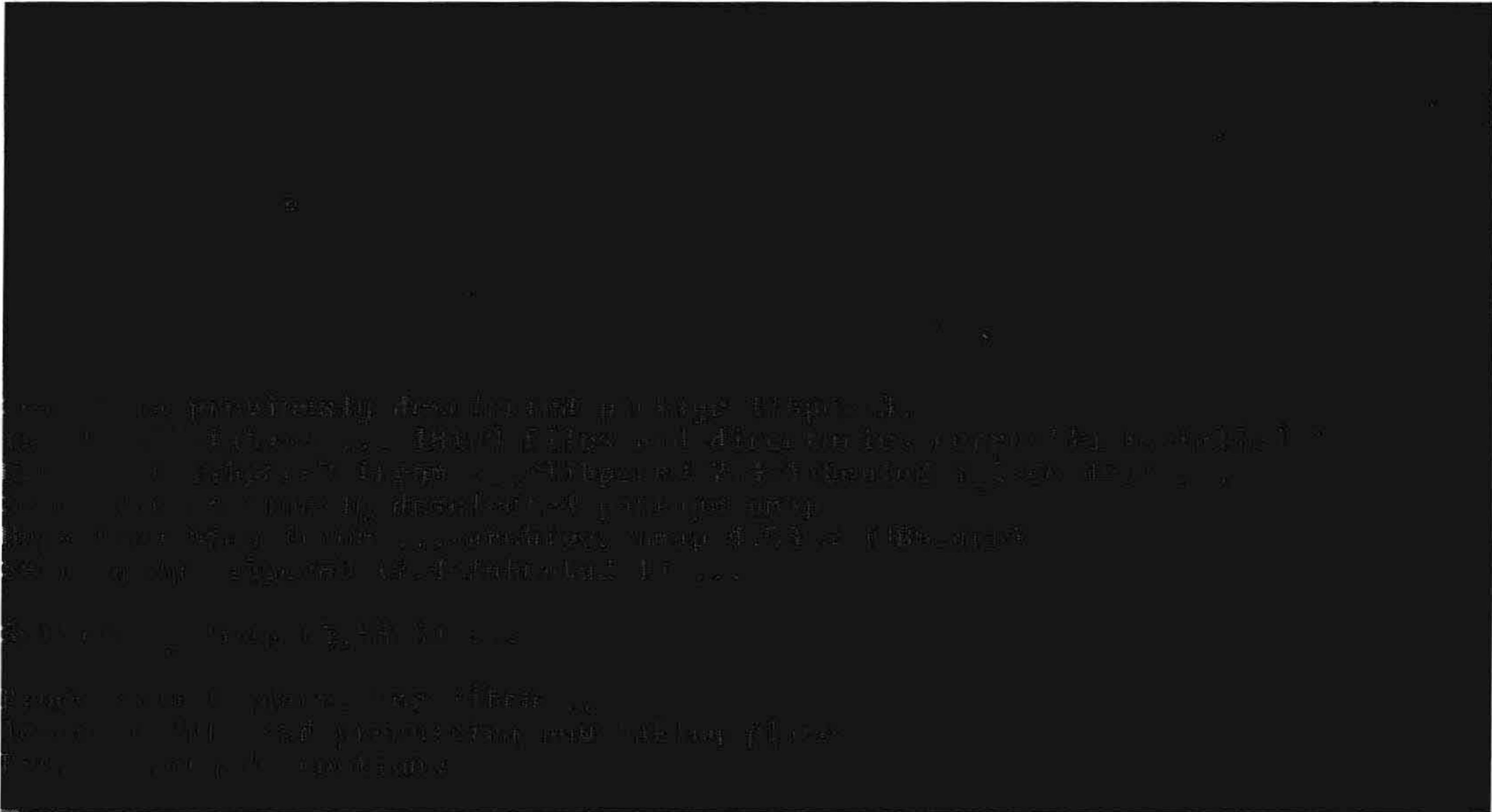


图 7-30 执行安装任务

安装结束之后，按回车键，回到 **Aptitude** 菜单。现在卸载刚刚安装的 **nmap** 软件包，以此来学习如何卸载软件包。再次用正斜线 “/” 找到 **nmap** 软件包，输入连字号 “-” 标记将要被卸载的软件包。你可以看到目标软件包的状态符号由 **i** 变成 **id**。如果我们对目标软件包设

置了自定义的配置文件，并希望在该步骤把这些配置文件也一并删除，则输入下划线“_”标记该软件包；然后，目标软件包的状态符号变成 p。如图 7-31 所示，nmap 软件包被标记为 id，意味着该软件包将被卸载。输入 g，提交所设置的一些列更改。

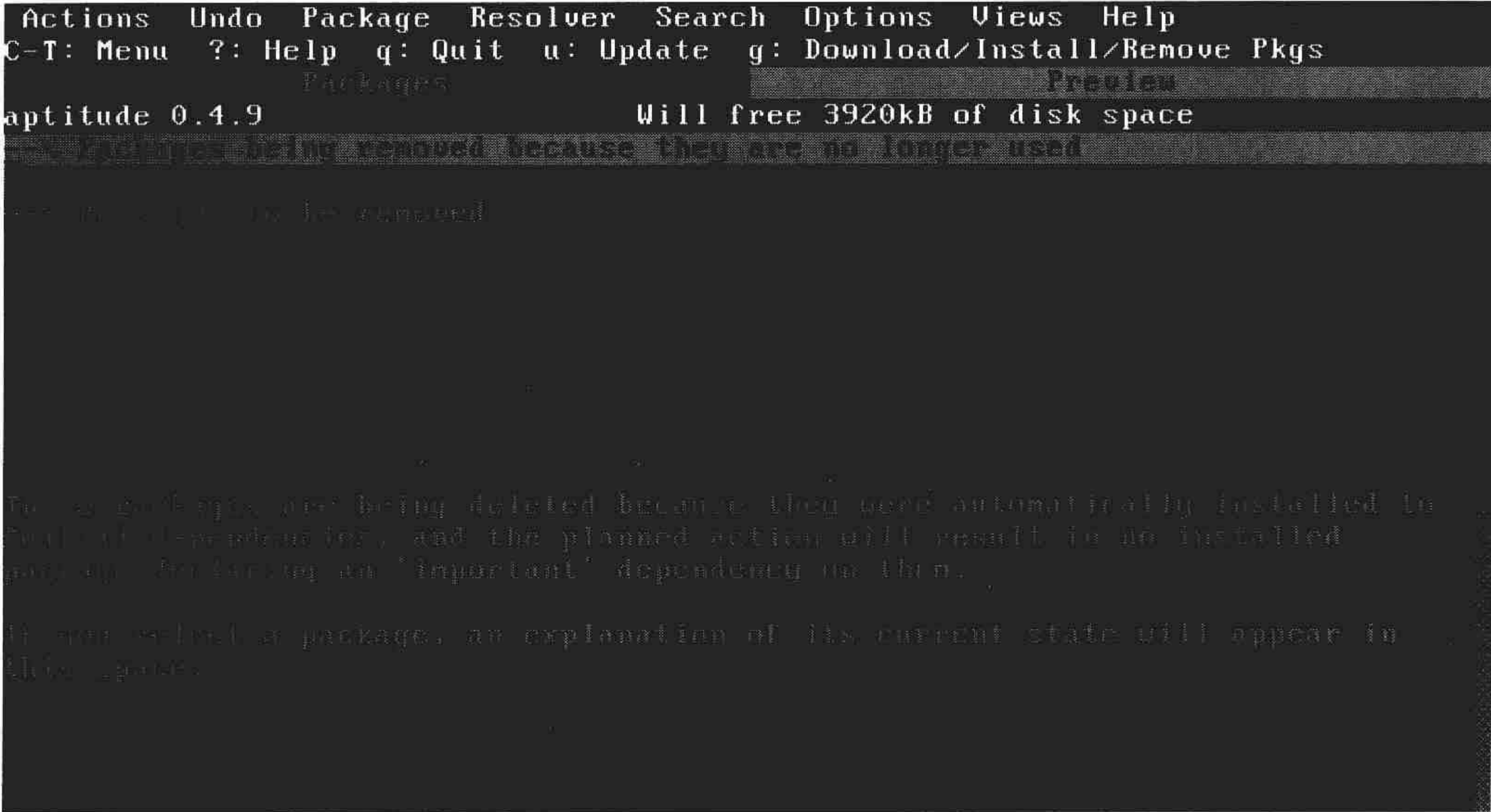


图 7-31 通过 aptitude 命令卸载软件包

因为并没有其他的软件包依赖于 libpcrc3 软件包，所以和当初的自动安装一样，该软件包将被自动卸载（见图 7-32）。再次输入 g，确认并执行上述改动。

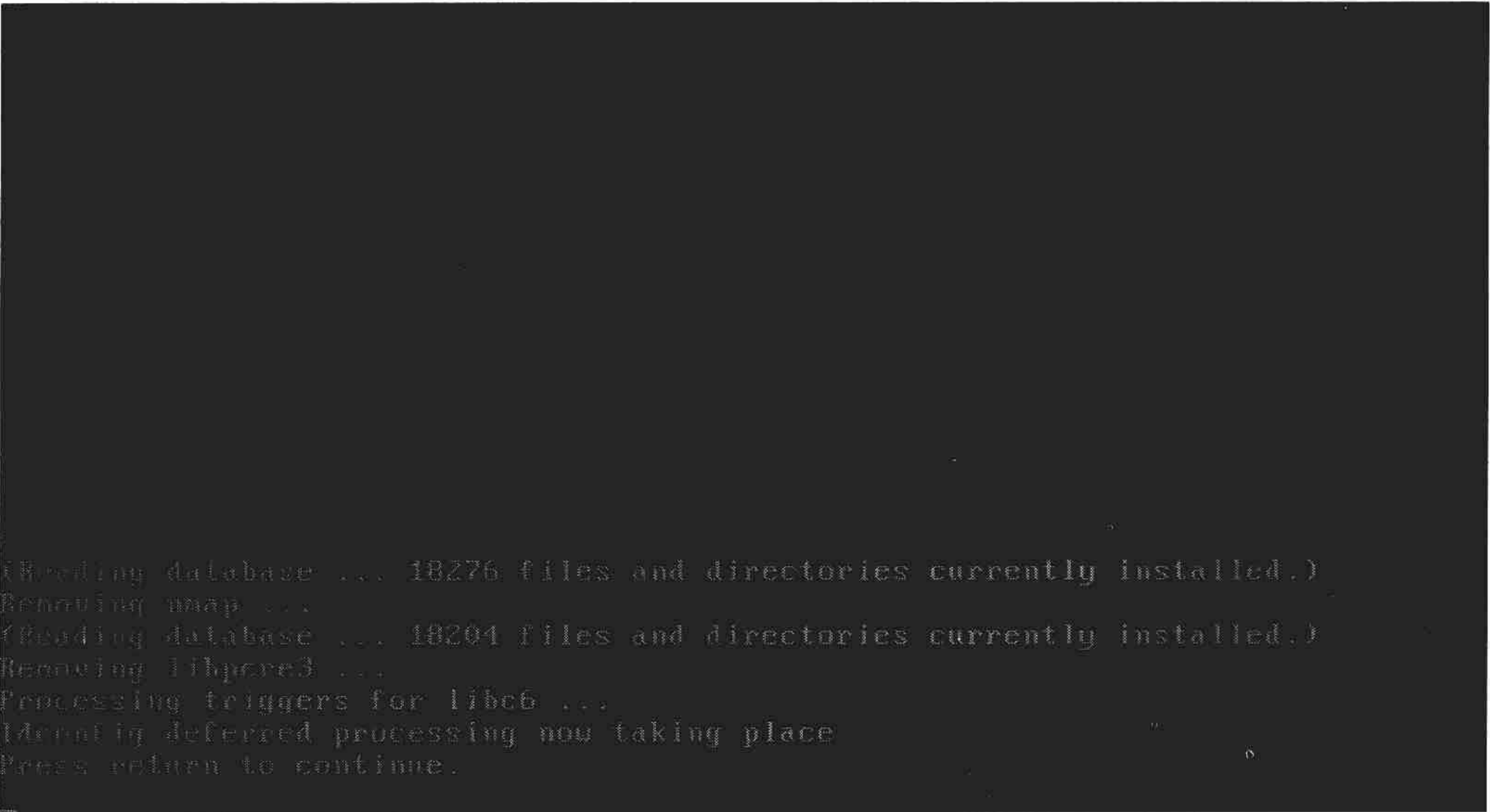


图 7-32 执行卸载操作

如果想了解更多的 aptitude 命令，可输入“?”键。图 7-33 列出了可用的 aptitude 命令集。

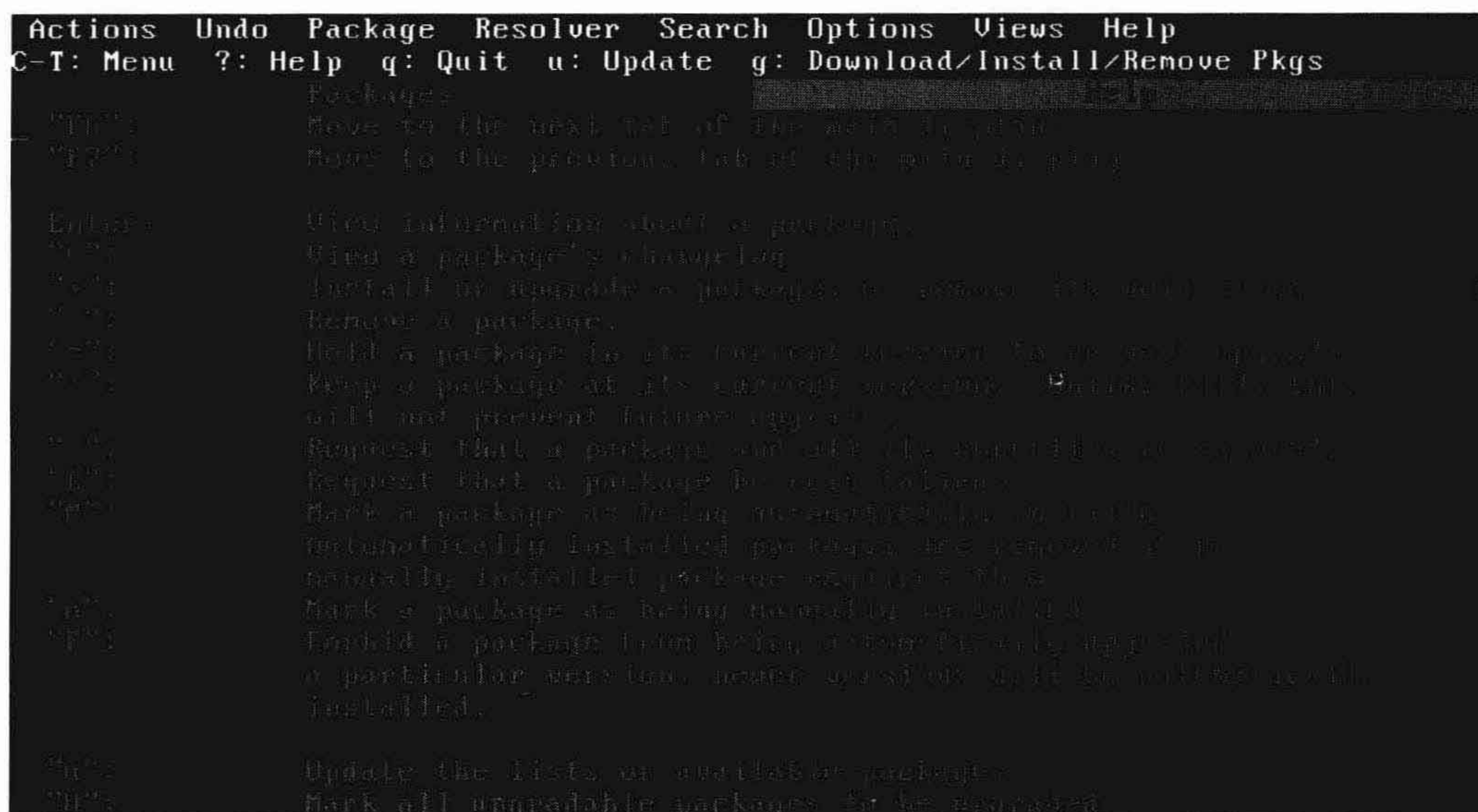


图 7-33 aptitude 命令列表

输入 **q** 可关闭上述的命令帮助列表。再次输入 **q** 则可以关闭 **aptitude** 工具。

1. 非交互式模式

虽然前面的导航选项与窗口选项都是用户友好的，但是这样的操作流程并不够快。为了使操作过程更快，**aptitude** 还提供了命令行操作模式，在该模式下，无须使用交互式菜单，只需执行带有软件包名字的参数命令即可。为了说明在命令行模式下使用 **aptitude** 与在 GUI 界面下一样便捷，我们仍然以安装 **nmap** 为例。在使用 **aptitude** 非交互式模式的时候，你必须以 **root** 用户身份进行软件包的安装和卸载。请看列表 7-17 所示的 **nmap** 的安装过程。

列表 7-17 使用 aptitude 非交互式模式安装 nmap

```
$ sudo aptitude install nmap
Reading package lists... Done
<snip>
The following NEW packages will be automatically installed:
  libpcre3
The following NEW packages will be installed:
  libpcre3 nmap
0 packages upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 0B/1218kB of archives. After unpacking 3920kB will be used.
Do you want to continue? [Y/n/?] y
Writing extended state information... Done
Selecting previously deselected package libpcre3.
(Reading database ... 18189 files and directories currently installed.)
Unpacking libpcre3 (from .../libpcre3_7.7-1ubuntu2.1_i386.deb) ...
Selecting previously deselected package nmap.
Unpacking nmap (from .../archives/nmap_4.53-3_i386.deb) ...
```



```
Setting up libpcrc3 (7.7-lubuntu2.1)...
Setting up nmap (4.53-3)...
Processing triggers for libc6 ...
<snip>
Writing extended state information... Done
Building tag database... Done
```

首先，`aptitude` 命令查看软件包的数据库，以确定可以从系统上卸载该软件包；如果软件包的某部分已被设置或挂起，那么 `aptitude` 命令将通知我们并中断卸载过程。

接着，`aptitude` 命令告知我们 `nmap` 软件包所安装的依赖软件和这些软件所占用的磁盘空间大小。注意，`aptitude` 命令会告知它需要 128KB 空间中的 0KB。因为之前已经安装过 `nmap` 软件包，所以原始的安装包还在主机上。现在，我们可以执行下列命令，确保 `nmap` 软件包已被安装。

```
$ nmap -v
Starting Nmap 4.53 ( http://insecure.org ) at 2008-08-31 16:05 EST
```

APT 缓存

当通过因特网安装软件时，这些软件包第一次被下载到主机时会被保存到位于 `/var/cache/apt/archives` 的缓存下。卸载某软件后，然后再次安装它，就无需下载安装包了。这些功能允许我们在多主机之间共享这样一个缓存，它在缓慢而昂贵的网络环境下很有用。另外两个例子是 `apt-cacher` 和 `apt-proxy`，它们都含有可用的软件包。

输入 Y 或者按回车键开始安装软件。`aptitude` 命令在内部使用 `dpkg` 命令控制安装过程。安装完毕后，`aptitude` 命令执行软件包所设置的触发器，设置并检查软件包的状态数据库，以确保每一个环节都成功完成。然后，我们执行 `nmap` 命令检查该软件包是否已被正确安装。

2. 使用 aptitude 卸载软件包

列表 7-18 给出如何使用 `aptitude` 命令卸载软件包。就像使用 `dpkg` 命令一样，我们可以再一次卸载或清空一个软件包。卸载意味着把除配置文件以外的所有相关信息从主机上删除，而清空则意味着删除整个软件包。

列表 7-18 使用 aptitude 删除软件包

```
$ sudo aptitude remove nmap
Reading package lists... Done
<snip>
The following packages are unused and will be REMOVED:
  libpcrc3
The following packages will be REMOVED:
  nmap
0 packages upgraded, 0 newly installed, 2 to remove and 0 not upgraded.
Need to get 0B of archives. After unpacking 3920kB will be freed.
Do you want to continue? [Y/n/?] y
<snip>
Writing extended state information... Done
Building tag database... Done
```


如果需要删除所有配置文件，你可以为 aptitude 命令添加附加参数--purge。

```
$ sudo aptitude remove --purge nmap
```

软件源

APT (Advanced Packaging Tool) 和 Aptitude 源码包都来自于在线软件源。如前所述，源是个软件包的集合，其中的软件包都是由其维护者维护。即将介绍的这两个软件包工具都使用了特定的配置文件 (APT 源文件) 来定义从哪里能获取在线软件包和需要哪种类型的软件包。

APT 和 Aptitude 工具使用这些源文件寻找成千上万的包信息。源的默认配置信息通常存储在 /etc/apt/sources.list 文件中，该文件在安装系统时候就已经被创建了。我们还可以把更多的源信息存放在 /etc/apt/sources.list.d/ 目录下。

简而言之，不同的源对应着不同的发布版本，而对发布版本而言，每个发布版本的版本号又对应着不同的源。一个更大的源集合则对应着不同类型的软件包。听起来是不是很复杂？其实，当你理解了就不觉得复杂了。以 Ubuntu 8.04 (代号为 Hardy Heron) 为例。如果安装了该版本的 Ubuntu，你可以在文件 /etc/apt/sources.list 中看到如下一行。

```
deb http://archive.ubuntu.com/ubuntu hardy main restricted universe multiverse
```

源的定义从资源类型和一个 URL 开始。资源类型指明源所包含的软件包是二进制文件还是源文件。deb 类型的源包含的是二进制软件包，而 deb-src 类型的源则包含应用程序的源文件。通常并不需要使用 deb-src 类型的源，除非你想要创建 Backports (请看补充知识“Ubuntu Backports”)。URL 指明提供软件源的服务器地址。下一个字段是发布代号，Ubuntu 8.04 的代号是“hardy”。我们可以在 http://en.wikipedia.org/wiki/History_of_Ubuntu_releases 上找到 Ubuntu 的其他发布代号。最后是一个列表，每个列表项设置了可用的软件包集合。

软件包被证书类型和支持级别分成几个部分。Ubuntu 有 4 个部分。第一个部分是“main”，它含有所有由 Ubuntu 开发者及标准组织提供的免费软件。不是直接由标准组织提供，而是由广大 Linux 社区提供的软件包属于“universe”部分。由标准组织提供的专利 (闭源但是免费——请看 http://en.wikipedia.org/wiki/Free_as_in_beer#beer) 软件属于“restricted”部分。最后，涉及专利和法律问题的软件——例如 MP3 播放器或者 DVD 播放器软件——属于“multiverse”部分。

如果只指定某一个部分，你可以限制系统能够安装哪种类型的软件包。例如，你可以指定只安装来自标准组织的免费软件，那么源文件可以如下这么写。

```
deb http://archive.ubuntu.com/ubuntu hardy main
```

作为练习，你可以打开网页 <http://archive.ubuntu.com/ubuntu> 去查看一个源是什么样子的。你可以从以下网页获知更多关于软件源及如何配置它们的信息。

<https://help.ubuntu.com/community/Repositories/Ubuntu>。

3. 使用 Aptitude 更新软件包

另一个标准的升级任务是通过第一次更新可用软件包列表 (检查是否拥有源中软件包的最新版本) 和执行升级操作来完成的。

列表 7-19 说明了如何执行一次可用软件包列表的更新操作。其实所进行的操作就是：aptitude 程序利用文件 /etc/apt/sources.list 中的源 (以及 /etc/apt/sources.list.d/ 目录中的附加软件源) 列表生成一个可用的软件包列表。

■提示：补充知识“软件源”里讨论了更多关于源的细节。

列表 7-19 aptitude 更新

```
$ sudo aptitude update
```

现在，在更新软件的时候，你可以选择两个更新方式：safe-upgrade 方式和 full-upgrade 方式。safe-upgrade 更新方式是在更新软件之前不删除原软件，有些时候该更新方式是有用的。有时在二次更新软件的时候，并不希望删除第三方软件包，我们使用 safe-upgrade 更新方式，第三方软件包不必更新到新的版本。但在某些情况下，就必须采用 full-upgrade 更新方式了，该方式通过先删除后安装所有必需的软件包，从而完全更新已安装软件。

列表 7-20 展示了每一种更新方式的命令用法。

列表 7-20 自动安装软件包的升级版

```
$ sudo aptitude safe-upgrade
$ sudo aptitude full-upgrade
```

高级软件包管理工具

在 Aptitude 出现以前，高级软件包管理工具（APT）面向基于 deb 类型的软件包提供了大部分的在线软件包管理功能。这些工具现在还有，但是它们所提供的某些功能已不再被 Aptitude 所支持，而且这些功能也不太好掌握，所以 Aptitude 收购了 APT 并兼容这些功能。由 APT 组合而成的命令包括 apt-get、apt-cache 和 apt-file。apt-get 命令负责下载、安装和卸载软件包及源文件。apt-cache 命令负责搜索软件包列表和显示软件包信息。apt-file 负责搜索软件包所提供的文件内容列表。

aptitude 工具一般指代 apt-get 工具，所以在本文的所有例子里我们并不使用 aptitude 命令，而是使用 apt-get 命令。但是，在解析软件包的依赖与冲突的时候，aptitude 具有更多的高级功能，所以建议读者尽可能地使用 aptitude 命令。例如，如果要使用 apt-get 命令来安装 nmap 软件包，你可以采用如下命令。

```
$ sudo apt-get install nmap
```

要想使用 apt-cache 命令找到某个特定软件包的信息，你可以采用如下做法。

```
$ sudo apt-cache showpkg nmap
```

要想使用 apt-file 命令找到某个属于一个软件包却并未随软件包安装在主机上的文件，你可以使用 apt-file 命令搜索由源提供的该软件包的文件集合。

```
$ apt-file search /usr/sbin/foo
```

上述操作需要列出系统上最新的目录。如果这些目录过期了，则可以通过以下命令把它们更新到最新版本。

```
$ sudo apt-file update
```

另外一种替代方法是可以在 <http://packages.ubuntu.com> 上搜索在线的软件包列表。

7.4 使用 Synaptic 进行包管理

如果你已经在 Ubuntu 系统中安装了 X Window 系统和 Gnome 程序，就能以另一种方式管理软件的安装。在 Ubuntu 系统中安装 `gnome-app-install` 和 `synaptic` 中的任何一个程序，都能允许你添加软件、卸载软件和管理软件源，而且这些操作不需要使用命令程序。然而，`synaptic` 本身只是一个后台使用 `apt` 和 `dpkg` 来完成所有包操作的前端程序（或者说包装）。

如果你不想在服务器上安装 X Window 系统可以跳过该部分。大多数服务器被放置在机柜中，并没有屏幕和键盘，因此安装一个庞大而又复杂的桌面环境根本没有用处。如果你想添加一个完整的 GUI 程序，可以通过安装 `ubuntu-desktop` 包或者使用 `aptitude` 安装 `kubuntu-desktop` 包来安装分别带有 Gnome 或者 KDE 的 X Window 系统。

当安装完成后，通过 `sudo /etc/init.d/gdm start` 或者 `sudo /etc/init.d/kdm start` 开启登录管理器并使用你的用户名和密码登入。

你可以通过“系统→管理”菜单来开启 `synaptic` 程序，如图 7-34 所示。



图 7-34 打开 Synaptic 包管理器

因为 `synaptic` 需要由 `root` 用户启动，所以需要使用 `sudo` 命令启动并且要求用户输入密码。第一次启动它，将弹出一个带有一些帮助信息的对话框。

该应用程序的布局简明。左侧有一个和在 `aptitude` 软件中看到的相应部分相同的列表。在列表下方是 5 个按钮，这些按钮允许你显示以不同的方式过滤出来的包。轮流单击这 5 个按钮就可以看到这些部分的列表是如何变化的。

7.4.1 添加软件

这次让我们通过 synaptic 再安装一次 nmap 软件。滚动那个长长的包列表以找到 nmap。根据位于底部的状态栏中的信息——24948 个包，可能需要一段时间才能找到。我们也可以单击“搜索”按钮并在“搜索”区域键入“nmap”来找到它（见图 7-35）。值得注意的是，通过“Look in”列表，你不仅仅可以通过“名字和描述”域来搜索。现在选择“描述和名字”，然后再次单击“搜索”。

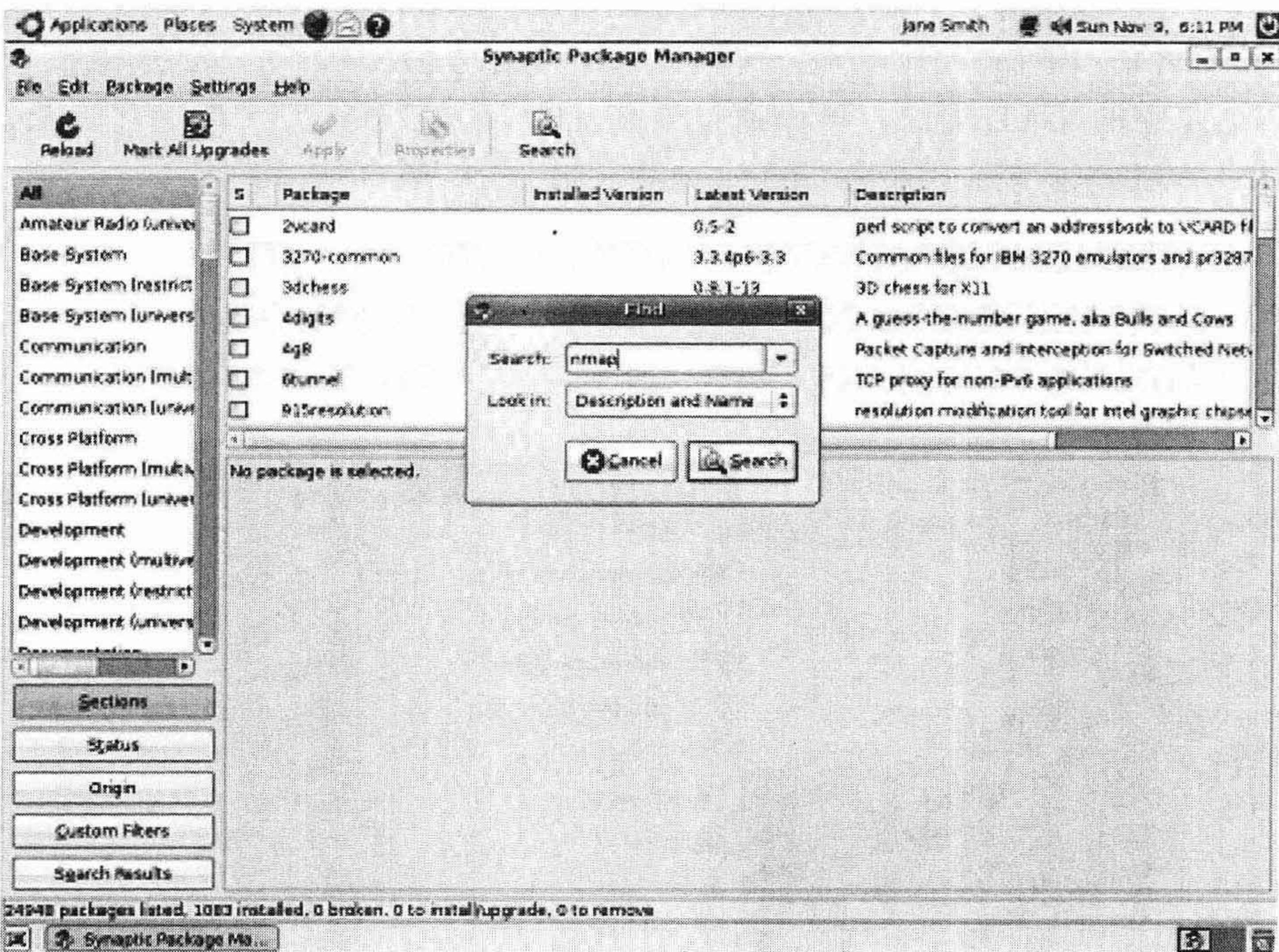


图 7-35 用 Synaptic 进行搜索

这样，synaptic 将列出所有在包的名字或者描述中出现“nmap”的包。你或许仍需要向下滚动列表以找到你要找的包。通过单击它的复选框并选择“安装”标签来将其添加到安装队列。就像 aptitude 一样，synaptic 将创建一个变化的内部列表。直到你单击“应用”按钮之后，这个列表才会被执行。因此你可以在一次操作中搜索所有你想要的软件并安装（或者卸载）它们。另外，你可以看到左侧的软件部分已经变为了被执行过的搜索操作的列表了。在任何时候进行搜索，synaptic 都会将其作为筛选器添加到这个列表中，这样你就可以快速地跳回到之前做的搜索操作。值得注意的是，这个列表只会保持到挂起的更改被应用之前。

单击“应用”按钮执行列表以安装 nmap。你将看到一个对话框告诉你本次操作中所有将要被安装、移除、升级或者阻止的包的信息（见图 7-36）。

再次单击“应用”按钮，进度框将告知你安装状态。当安装完成后，关闭对话框（见图 7-37）。



图 7-36 应用挂起的更改

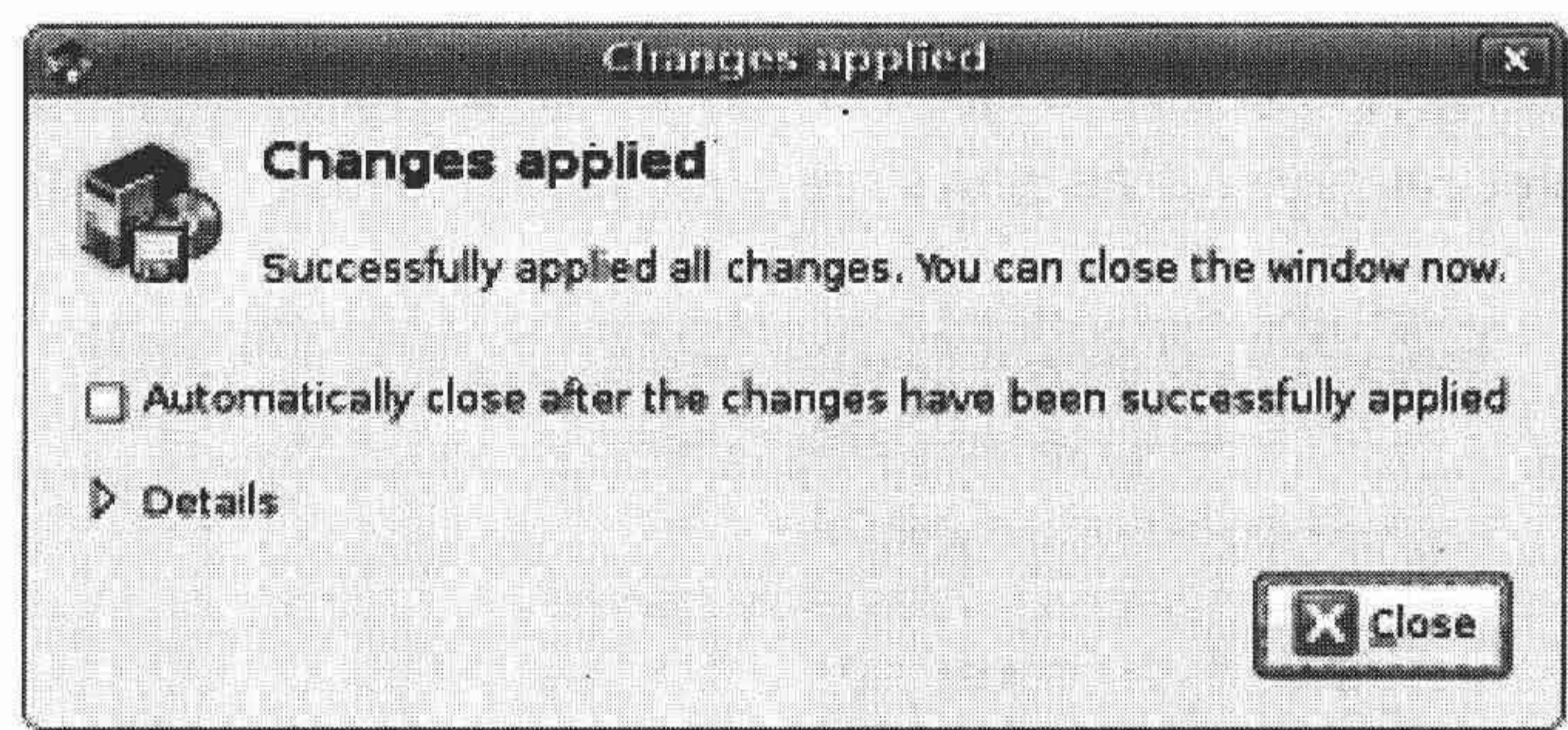


图 7-37 安装完成

像 aptitude 和 apt-get 一样，synaptic 将自动选择你将要安装的软件所需要的任何一个相关的包。

■注：因为 libpcrc3 包已经作为 X 和 Gnome 程序的依赖包被安装过了，nmap 应用程序这次不再需要安装它了。这意味着该包在我们卸载 nmap 时也不会被自动移除。

7.4.2 移除软件

当然，你也可以通过 synaptic 移除软件。左下角的筛选器按钮在这派上了用场——单击“状

态筛选器”按钮然后选择“已安装”状态。现在列表变为显示你的系统上已经安装的所有的包。

单击靠近 apt 包的黑底的复选框并选择“Mark for removal”。完成后，你会看到另一个“Mark for complete removal”选项。后者和 dpkg 中的“--purge”选项是等价的。在将 apt 标记为删除之后，单击“应用”按钮。

Synaptic 将抛出一个警告（见图 7-38），以通知你移除这个包可能会导致系统不可用。你一定不想这样做，所以单击“No”。Synaptic 有一些基本的保护措施来试图阻止你做一些你一般不应该做的事情。

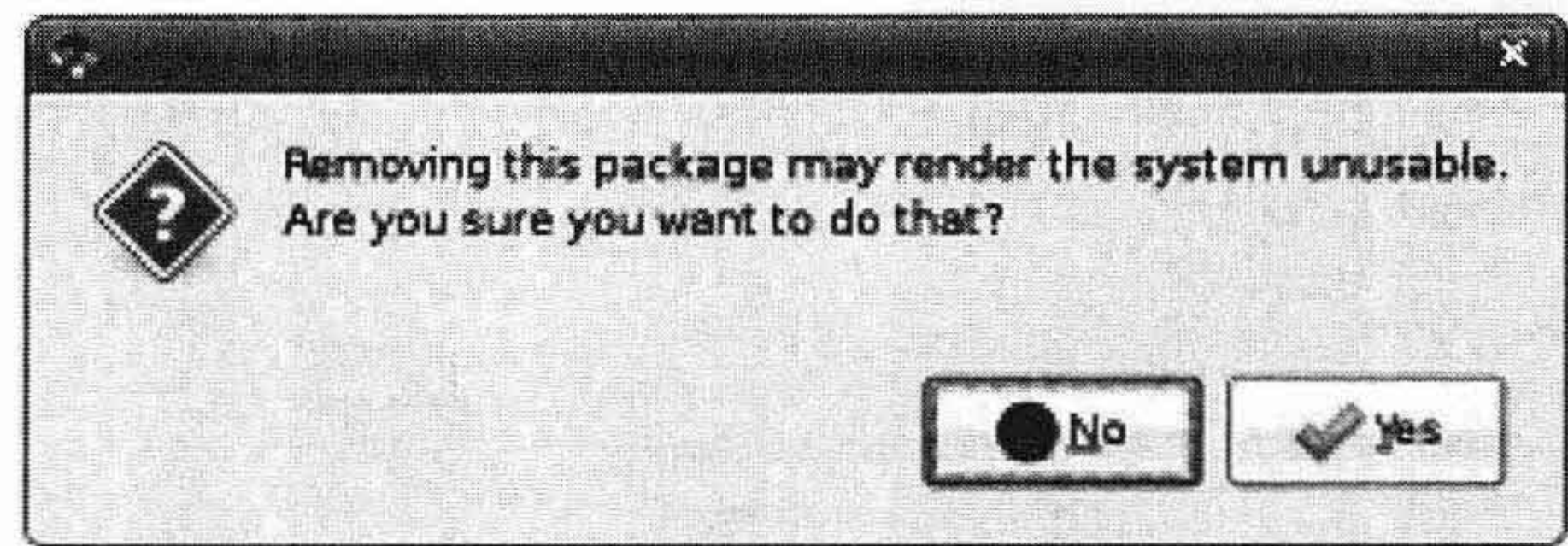


图 7-38 Synaptic 的挽救警告

在已安装软件列表中找到 nmap，把它标记为完全移除，然后单击“应用”以移除它。

7.4.3 管理软件源

比起手动管理 /etc/apt/sources.list 文件，Synaptic 提供给你一个易于使用的接口以允许、禁用或者改变程序源。要访问它，你可以选择“设置”菜单里的“源”选项。你也可以在 Gnome 主菜单栏中选择“系统→管理→软件源”来访问这个对话框。

在打开的“Ubuntu 软件”标签上，你可以选择一个官方镜像服务器来使用，并可以规定在你的系统中想要得到哪些软件部件（见图 7-39）。除非你有特殊的原因，否则最好选中所有的复选框。

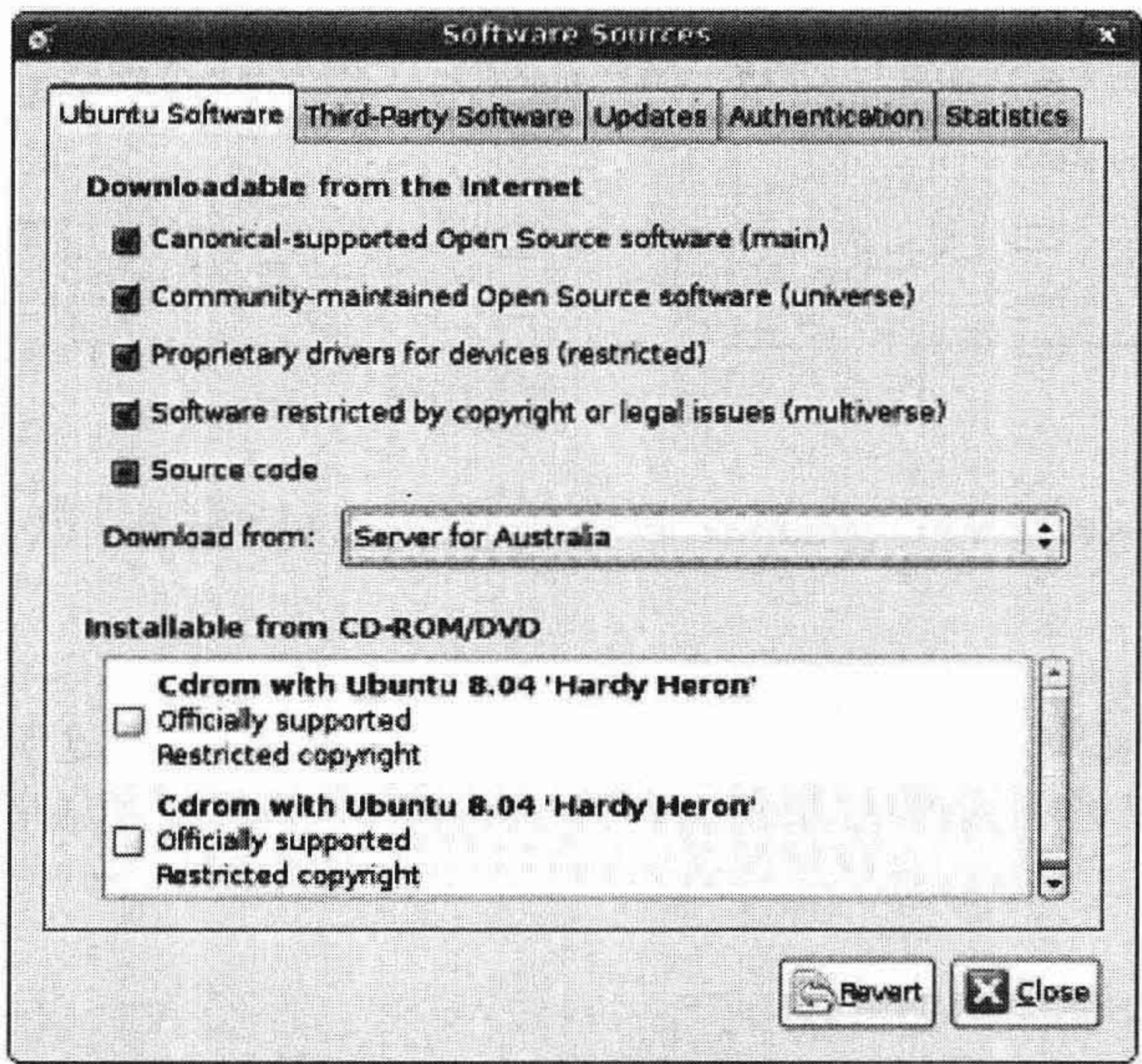


图 7-39 改变软件源设置

在第二个标签（“第三方软件”）里你可以添加额外的软件源。这些额外的源通常提供给你一个 deb 行，你可以简单地将其粘贴进对话框并添加到你的源集合中（见图 7-40）。



图 7-40 添加一个第三方源

在你添加这一行之后，你可以编辑它，并允许添加一个个性化的描述，如图 7-41 所示。值得注意的是，这里的例子是一个并不存在的源，因此在关闭 synaptic 之前，将要再一次的删除它。

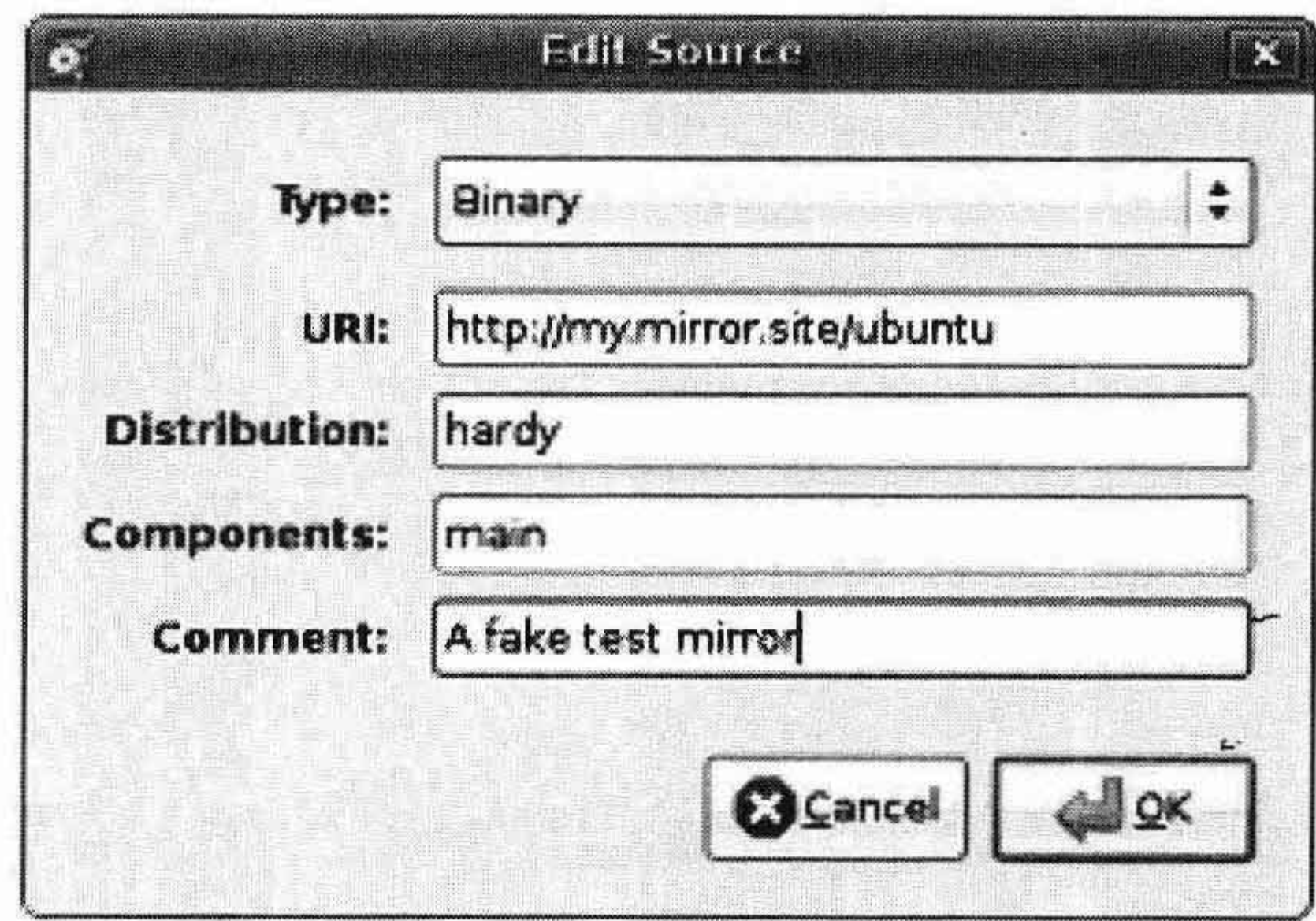


图 7-41 添加源的描述

软件的更新通常在它们自己的源里工作，因此在“更新”标签中保持一个生产服务器上的关键性更新和推荐更新处于可用状态显得尤为重要。

另两个标签是“验证”和“统计”。第一个允许你引入或者移除添加的某些源可能会用到的密钥，就像在命令行界面中 `apt-key` 命令所做的一样。这些密钥是验证你安装的包可信的重要方式。当选中“统计”选项后，运行 `popularity-contest` 程序将已安装包的列表提交给 Ubuntu 项目。该功能将汇总来自所有其他 Ubuntu 用户的提交信息，并用来决定哪些应用应该作为默认程序和系统一起发布。

你可以通过<http://popcon.ubuntu.com> 查看收集的统计信息。

7.4.4 更新与升级

当你关闭软件库对话框时，Synaptic 会发出软件库已被改变的警告并要求更新它内部的软件包列表。单击“Accept”按钮，然后再单击“Reload”按钮。将会出现如图 7-42 所示的一个对话框实时通知你下载的状态。

就像 `aptitude update` 一样，Synaptic 会链接所有的软件库并下载所有可用软件包的最新列表。这些更新列表中，很有可能包括已安装软件包的更新。要弄清楚包含哪些类型的更新，单击“Status”（状态）过滤按钮，然后选择“Installed (upgradeable)”项进行过滤。（见图 7-43）。

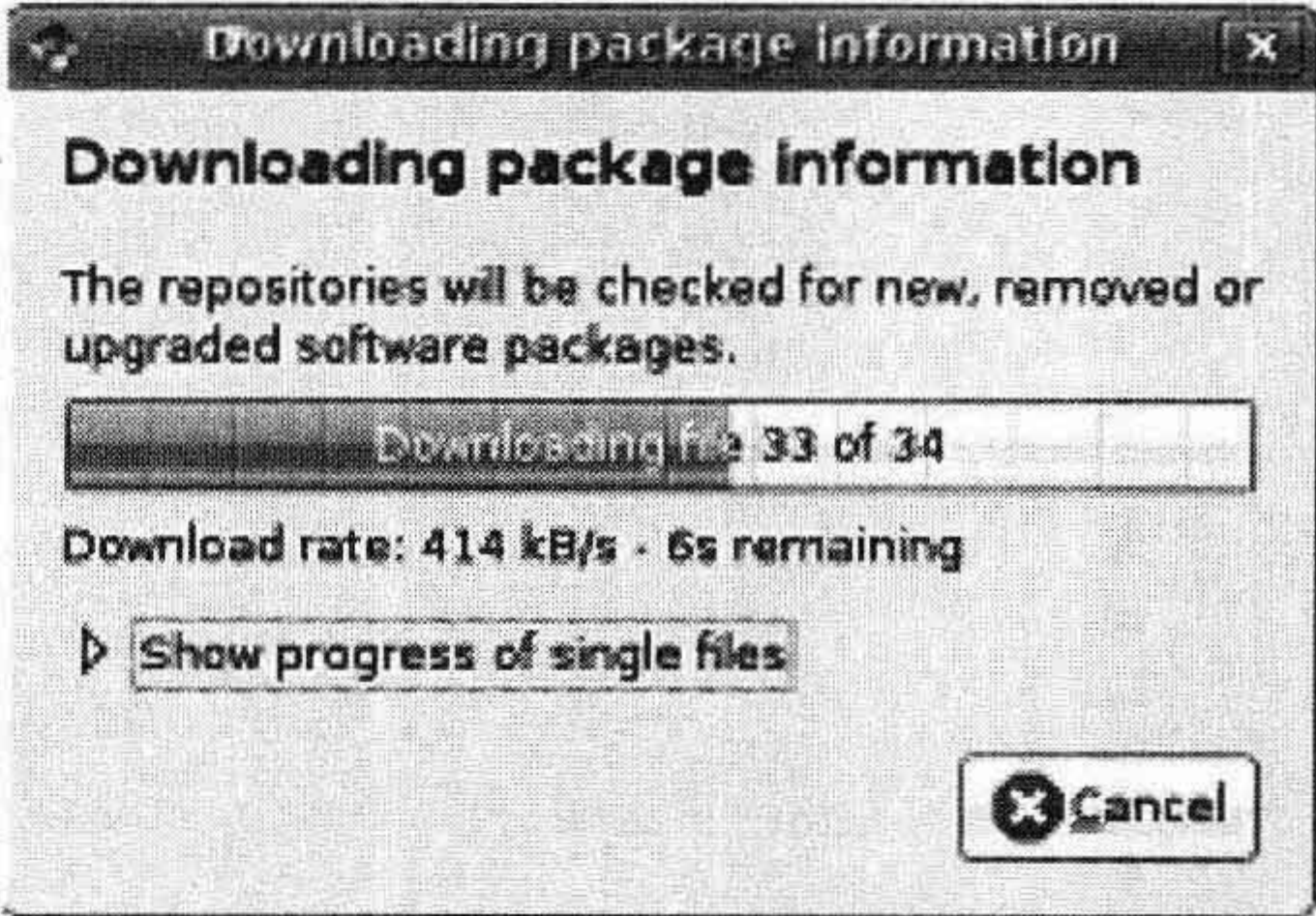


图 7-42 更新软件包列表

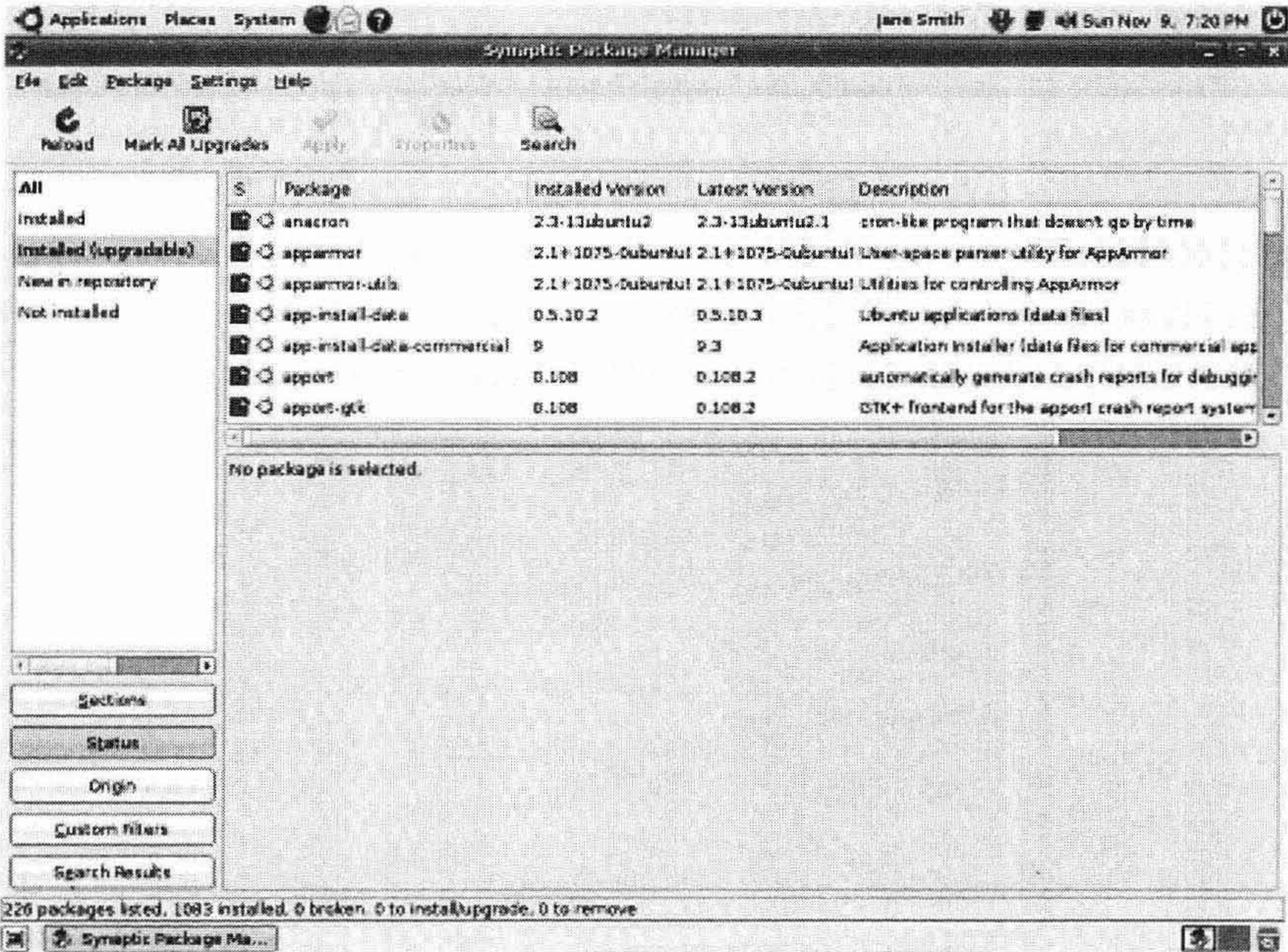


图 7-43 查看可用的软件更新

我们建议你安装所有可用的更新。保持你的软件处于最新状态意味着你的主机将不那么容易受到错误和恶意用户的影响。要安装更新，首先单击“Mark All Upgrades”，然后单击“Mark”。Synaptic 就会添加所有这些更新到它的内部任务列表。要执行这个列表，单击“Apply”按钮，确定要更改的列表。

7.4.5 使用更新管理器

即使 Synaptic 当前没有运行，Ubuntu 也会通过顶部面板上的一个红色小箭头图标通知你是否存在可用更新。（见图 7-44）。

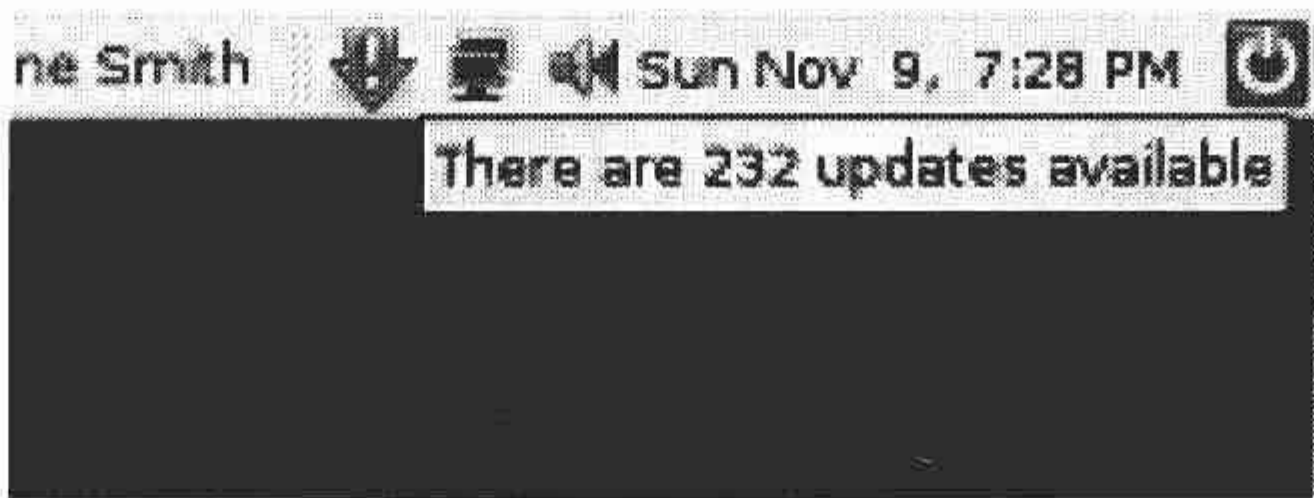


图 7-44 可用更新通知

双击箭头图标将打开更新管理器，它列出了所有可用更新（见图 7-45）。

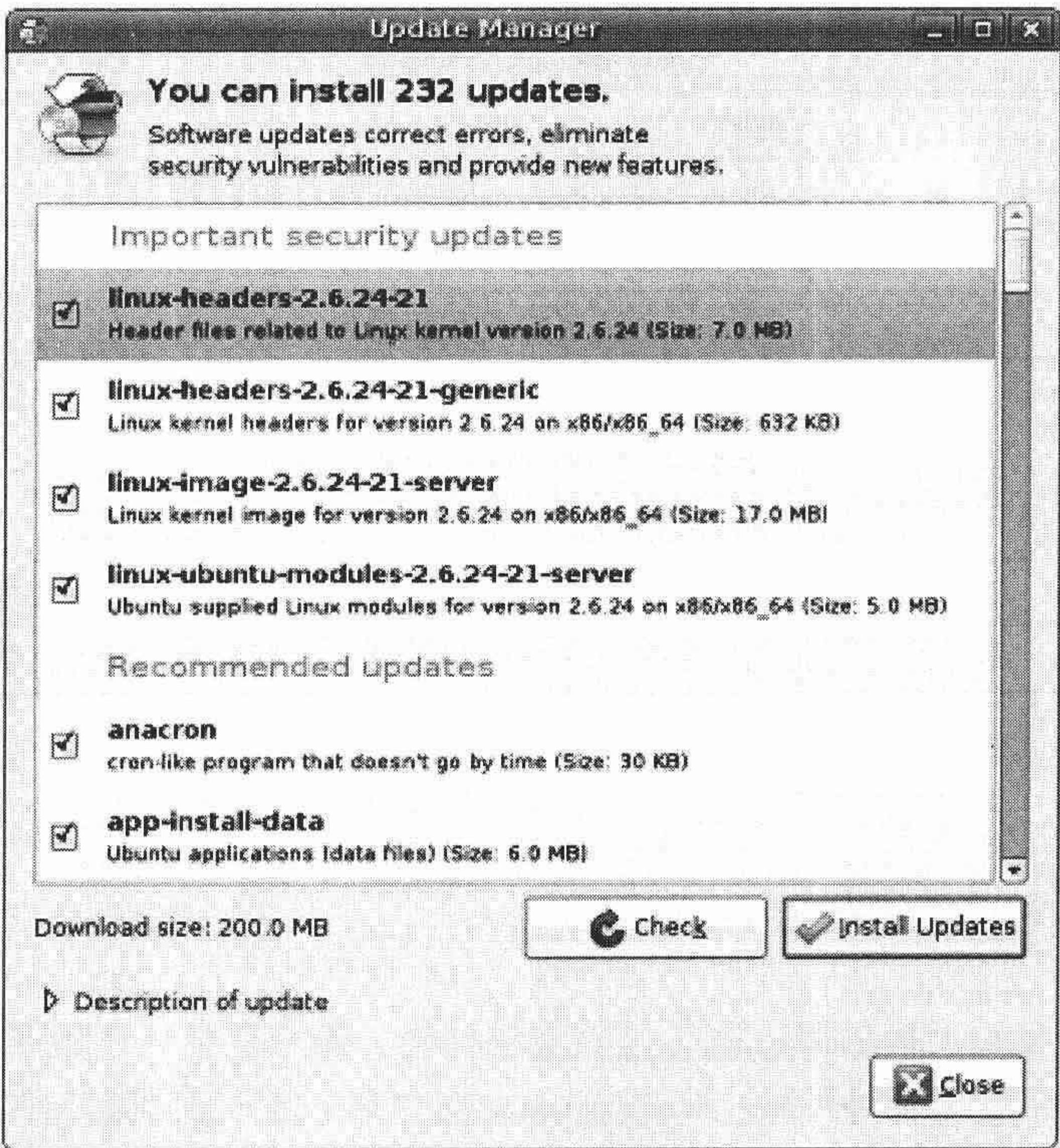


图 7-45 更新管理器

当你单击“Install updates”时，更新管理器将启动一个 Synaptic 软件，Synaptic 将会在下载和安装更新包之前询问你的密码。

7.5 使用 dpkg 包管理

Ubuntu 上面最基本的软件包管理工具是 dpkg（音为“dee-package”）。APT 和 Aptitude 工具都是以 dpkg 为核的，就像 Yum（软件包管理器，全称 Yellow dog Updater, Modified）是以 rpm 命令为内核的一样。

■提示：我们建议你使用 Aptitude 工具而不是 dpkg 来管理你的软件包。Aptitude 会考虑并管理各软件包之间的相互依赖关系，但这是 dpkg 无法做到的。

dpkg 命令允许你选择要安装的软件包，你可以安装以前下载的软件包并且查询它们属于系统中的哪个软件包。通过执行 man dpkg 命令，你可以找出 dpkg 所有的命令选项（并且有很多）。如列表 7-21 所示。

列表 7-21 dpkg 帮助页
\$ man dpkg

表 7-5 列出了一些主要的命令选项和参数。

表 7-5 dpkg 命令选项和参数	
选项和参数	描述
-l --list	列出主机上所有已安装的软件包信息
-p --print-avail	打印软件包信息
-c --contents	列出一个特定的卸载包的内容
-L --listfiles	列出已安装软件包的内容
-i --install	在你的主机上安装一个软件包
-s --search	查询系统中的某个安装文件属于哪个软件包
-r --remove	卸载你主机上的一个软件包。保留相关配置文件
-P --purge	清除你主机上的一个软件包。删除相关配置文件

首先，我们将获得一个列表，它列出了我们新的 Ubuntu 系统中已经安装的软件包。如列表 7-22 所示。

列表 7-22 已安装软件包列表
\$ dpkg -l

该命令将产生我们主机上所有软件包的完整列表。通常这是一个很长的表单，但是我们可以通过管道命令将 dpkg 命令的输出传送给 more 命令来一页一页地显示结果（见列表 7-23）。然后你可以在任何想要的时候翻页查看。

列表 7-23 dpkg -l 与 more 命令配合使用
\$ dpkg -l | more

注意列表 7-24 中 dpkg 列表分 4 项输出：状态（status）、名称（name）、版本（version）

以及描述（description）。

列表 7-24 命令 dpkg -l 输出结果

```
$ dpkg -l | more
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-f/Unpacked/Failed-cfg/Half-inst/t-await/T-pend
|/Err?=(none)/Hold/Reinst-required/X=both-problems (status,Err: uppercase=bad)
|/-Name          Version          Description
+++-----
ii adduser        3.105ubuntu1      add and remove users and groups
ii apache2        2.2.8-1ubuntu0.3  Next generation, scalable, ➡
extendable Web se
```

■注：虽然没有自动地从列表 7-24 中清除，最开始三行实际上输出的是状态项。

让我们花一些时间看看表中的这几栏，状态栏实际上包含了软件包可以处在的三种状态。

- 预期状态。
- 当前状态。
- 错误。

通常情况下，状态将为 ii，即软件包已被安装并且如果存在更新版本的话，软件包将被更新。表 7-6 列出了最常见的状态标号和它们的含义。

表 7-6 dpkg 的状态标号	
标号	含义
ii	软件包已安装并安装可用更新
hi	软件包已被搁置和安装但不会安装更新
un	软件包未安装
rc	软件包未安装，但存在残留的配置文件（通常这意味着安装包已被卸载）

其他栏的内容就不言而喻了。它们包括软件包的名称、版本（如果软件包是刚刚安装的），以及一个简单的描述。

通常，你并不需要所有事物的列表，因此我们可以通过 dpkg 的一项匹配字符串来限制软件包输出的内容。Ubuntu 的 Linux 内核名为 linux-image，这和小红帽的内核 kernel 名称不同。列表 7-25 列出了通过一条通配符得到的我们主机上安装的所有 linux-image（这意味着我们可以通过 linux-image 中的字符串获得所有的结果）通过在我们要使用的字符串前面和后面加上符号*。

■提示：使用通配符是我们在使用字符串时一个很有用的方法，特别是文件名一类的字符串。你可以在以下网站阅读到关于通配符的内容，网址为 <http://www.faqs.org/docs/abs/HTML/globbingref.html>

在这种情况下，我们使用*linux-image*命令（通过使用 linux-image* 和 linux-image 作为你的搜索字符串，你将会对通配符有一定的了解）。

列表 7-25 Linux 内核列表

```
$ dpkg -l '*linux-image*'
Desired=Unknown/Install/Remove/Purge/Hold
```



```
| Status=Not/Installed/Config-f/Unpacked/Failed-cfg/Half-inst/t-await/T-pend
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
|// Name          Version          Description
+++-----
un  linux-image    <none>          (no description available)
un  linux-image-2.  <none>          (no description available)
ii  linux-image-2.   2.6.27-16.30    linux kernel image for version 2.6.24 on x86
ii  linux-image-se   2.6.24.16.18    linux kernel image on Server Equipment.
```

这里我们可以看到几个 `linux-image` 软件包，它们的名字都是基于它们所包含的内核版本的。这些如此不同的内核软件包的名字并不相同，从而允许你安装多个内核，此外，你会发现 `linux-image` 软件包，即所谓的虚拟软件包。这个软件包并不包含任何文件，但它包含一个到最新可用内核的链接，所以正常的升级总是包含所有可用的新的 Linux 内核包。

提示：由于默认 Linux 终端只有 80 个字符宽而 `dpkg` 命令要求在单行显示每个软件包的信息，所有如果软件包名称长度大于 14 个字符，那么将无法显示全称。对于很多软件包来说这不是问题，但是内核包的名字通常很长，因此不能完全显示。若要解决此问题，你可以改写 `dpkg` 的终端的显示宽度，通过使用环境变量可以使它显示更多的信息。要做到这一点，执行如下命令：`$ COLUMNS=200 dpkg -l '*linux-image*'。`这条命令告诉你的主机，你的屏幕宽度为 200 个字符，从而使它显示更多内容。

7.5.1 查看软件包细节

利用 `dpkg` 命令我们可以查看安装的 `kernel` 包的一些信息。列表 7-26 显示了利用 `dpkg -p` 命令对已安装的 `Linux-image` 程序包相关信息查询的结果。

列表 7-26 `dpkg -p` 命令的输出结果

```
$ dpkg -p linux-image-2.6.27-16-server
Package: linux-image-2.6.27-16-server
Priority: optional
Section: base
Installed-Size: 60524
Maintainer: Ubuntu Kernel Team <kernel-team@lists.ubuntu.com>
Architecture:i386
Source: linux
Version: 2.6.27-16.30
Provides: fuse-module, ivtv-modules, kvm-api-4, linux-image, linux-image-2.6,
redhat-cluster-modules
Depends: coreutils | fileutils (>= 4.0), initramfs-tools (>=0.36ubuntu6),
module-init-tools (>= 3.3-pre11-4ubuntu3)
Pre-Depends: dpkg (>= 1.10.24)
Recommends: lilo-(>= 19.1) | grub
Suggests: fdutils, linux-doc-2.6.24 | linux-source-2.6.24
Conflicts: hotplug (<< 0.0.20040105-1)
Size: 18447888
Description: Linux kernel image for version 2.6.24 on x86/x86_64
This package contains the Linux kernel image for version 2.6.24 on
x86/x86_64.
```

Also includes the corresponding `System.map` file, the modules built by the packager, and scripts that try to ensure that the system is not left in an

unbootable state after an update.

Supports Server processors.

Geared toward server systems.

You likely do not want to install this package directly. Instead, install The linux-server metapackage, which will ensure that upgrades work correctly, and that supporting packages are also installed.

除了提供描述和版本说明，每一个安装软件包还包含提供技术支持的 Email 和安装时所需要的磁盘空间信息。Depends 部分还详细说明了该安装包被安装前所需要安装的其他软件包——依赖包。而 Provides 部分显示了该软件包对其他软件包的功能上的支持。

一、查看软件包内容

除了能获取软件包的描述信息，你也可以查询安装了哪些文件以及安装的目标路径。如下所示，我们通过 `dpkg -l` 命令找出软件包 `nmap` 的内容。

```
$ dpkg -l nmap
```

该命令返回软件包 `nmap` 安装的文件完整列表。

■提示：列出安装到包含可执行程序目录的文件可以更快地找出软件包所提供的命令。我们输入 `dpkg` 列出软件包内容，通过 `grep` 检索查询的结果筛选出包含字符串 `bin` 的文件和文件目录，例如：`dpkg -L <package name> | grep bin`。

二、执行文件搜索

对于已经存在于主机上的文件，通过 `dpkg` 命令，你能确定它们属于哪个软件包，如下所示。

```
$ dpkg -S /usr/sbin/userdel
```

这条命令告诉你哪个软件包提供了 `userdel` 命令。

7.5.2 安装软件包

所有 Ubuntu 软件包都是由三部分组成：文件名称、文件版本和目标架构。例如，适于 32 位英特尔机器的 2.17 版的 `foobar` 软件包为 “`foobar-2.17_i386.deb`”。

■注：目标架构是指主机处理器采用的架构，如 `i386`（intel 80386）和 `X64`。

一旦获得适合你的 Ubuntu 版本和处理器架构的软件包时，就能通过 `dpkg` 命令进行安装。由于软件包包含的文件需要被安装在被保护的系统位置（如 `/bin`、`/usr/sbin` 等），安装过程需要在 `root` 用户权限下进行。在 Ubuntu 中，读者能用第 4 章中讨论的 `sudo` 命令使得安装过程获得 `root` 用户权限。

```
$ sudo dpkg -i nmap_4.53-3_i386.deb
```


在软件包的安装过程中，dpkg 命令可以查看其安装进度。对于不同的软件包，你可能需要回答一些问题，来对软件包进行配置。当然，不用担心，之后还能改变这些问题的答案。

安装手动下载的软件包的时候，一定注意。如果该软件包不兼容你当前的 Ubuntu 版本，那么它所需要的其他软件包可能不可用或者不是正确的版本。如果强制安装，可能出现“unresolvable dependency”（不可解决的依赖性）错误，这时候，软件包被部分安装，而在该问题被解决之前，它将阻止其他的软件包管理，使得你的软件包管理器变得不稳定。不过，dpkg 将会发出警告并阻止安装过程，除非被强制执行。一个最佳的做法是从第三方来源下载软件包之前，总是检查该软件包本身是不是在该系统版本上可用。

7.5.3 卸载软件包

有两个不同的方法卸载一个 ubuntu 软件包：一个只是卸载软件包本身，另一个是从系统中完全清除它。当执行--remove 操作时，只能删除主机中该软件包安装到系统中的文件，而不能删除被修改的配置文件。当执行--purge 操作时，dpkg 将删除该软件包安装的所有文件，当然也包括你修改的配置文件。

为什么需要两种方法？因为很多时候，你想完全删除所有文件，而有的时候，可能只是想删除部分文件，以便以后重新安装该软件包。

列表 7-27 列出了使用这两种 dpkg 命令删除主机上软件包的方法。

列表 7-27 删除软件包

```
$ sudo dpkg --remove nmap
或
$ sudo dpkg --purge nmap
```

7.6 编译源代码

尽管软件包数目很多，但并不是所有的都是可用的方便的 deb 软件包或者 RPM 软件包。如果一个软件不是可用的软件包格式，甚至不是发行版的源文件包（查看“Ubuntu backports”侧栏），你可能需要从源代码进行组建。

Ubuntu backports（发行版）

有时，如果读者需要的特定版本不可用，你就可能需要去创建一个发行版。这时候很可能是在用老版本的系统，或者还未封装你的系统版本对应的新版本应用程序。创建一个 backport（发行版），需要从新版的（或者旧版的）Ubuntu 系统获得源文件包，并且在你的机器上编译。发行版的问题已经超出了本书讨论的范围，但是网上有很多很优秀的参考资料，教你如何去用、如何去创建它们。我们推荐一个很不错的入门网站：<https://help.ubuntu.com/community/UbuntuBackports>。

在这个部分，我们将展示如何从源代码来编译软件，并且提供一些建议，告诉你如何使得这些源安装文件易于管理。

从源代码组建应用程序一般有以下三个步骤。

1. 配置程序。
2. 编译或者生成程序。
3. 安装程序。

下面我们将和读者一起去了解这 3 个阶段，并再次使用 `nmap` 程序作为例子。这一次，不是直接从软件包进行安装，而是我们组建截止到当前最新版本的 `nmap` 程序（4.76 版）。首先，我们使用下载工具 `wget` 从 <http://nmap.org/> 上获取程序的压缩包。

```
$ wget -c http://nmap.org/dist/nmap-4.76.tar.bz2
```

注：压缩包（通常被压缩过）是包含一系列文件或者/和文件夹的文件。创建这些压缩软件包的应用程序叫作 `tar`（Tape ARchive），这样命名是因为开始的时候它被用于在磁盘中进行文件归档。压缩包的文件扩展名，通常表明该文档是如何被压缩的。例如，文件扩展名 `tar.gz` 或者 `tgz` 表明采用 `gzip` 进行压缩，而用 `bzip` 压缩的文件，其文件扩展名是 `tar.bz2` 或者 `tbz`。通过查看手册，读者可以知道更多关于 `tar` 的信息。

注：`-c` 标志告诉 `wget` 恢复未完成的下载任务。这意味着当下载任务意外中断时，可以不必从头开始下载该文件。如果下载中途失败，你只用重新运行命令，并加上 `-c` 操作，即可从中断的地方恢复下载任务。

通过 `wget` 命令下载的文件，其保存路径是当前文件目录，这样我们就获得一个可供解压的压缩软件包。从文件扩展名可以看出，它是通过 `bzip2` 进行压缩的，这样我们需要添加 `-j` 标志，使 `tar` 采用 `bzip2` 对压缩包进行解压。

```
$ tar xvjf nmap-4.76.tar.bz2
```

提示：如果不知道压缩包采用的是什么压缩方法，你能通过 `file` 命令找出。运行 `file <tarball>`，文件系统将检查压缩文件的物理大小（magic bytes），这个可以说明文件的类型。成千上万的文件类型对应的物理大小被存储在 `/usr/share/file/magic` 的文件清单中。一旦知道压缩包是如何被压缩的，就能对它进行解压缩了。

`tar` 附加 `-v` 参数能够详细地显示从压缩包中解压的所有文件的路径。如果已经知道压缩软件包都包含哪些文件，你可以不添加这个参数，这样终端不会滚动显示这些冗余信息。不显示文件信息，额外的好处是解压操作将会更加快速。

压缩包解压之后，通过以下操作，我们可以进入源目录查看它包含了哪些文件。

```
$ cd nmap-4.76
~/nmap-4.76$ ls
```

许多情况下，我们可以通过 `README` 或者 `INSTALL` 文件查看有关程序编译和安装的说明。对于 `nmap` 程序，没有 `README` 文件，而是包含 `INSTALL` 文件。阅读这些文件，我们可以找到安装该程序的说明。

```
$ less INSTALL
```

从 `INSTALL` 文件中可以知道，安装说明文档不在压缩软件包中，而是在网上。我们可以参阅 <http://nmap.org/book/inst-source.html> 上 `nmap` 的在线文档，获取 `nmap` 的详细信息。参

照这些说明，我们就可以编译和安装程序了。

但是在编译程序之前，我们需要安装一个编译器和相应的库文件、工具。这些通常是封装在一起的，所以只需通过软件包管理器进行简单的安装即可。表 7-7 列出了 Red Hat 和 Ubuntu 编译时所需要的软件包。

表 7-7 安装编译和基本的组建工具

系统版本	命令
Red Hat	yum install gcc make
Ubuntu	aptitude install build-essential

7.6.1 配置

编译器安装完成后，我们可以对源代码进行配置，以用于应用程序的组建。大部分软件都是可配置的，不仅是在可用的功能方面，在安装位置方面也是如此。在组建一个应用程序之前，我们可以先使用 `configure` 命令配置它。如列表 7-28，我们通过运行 `configure` 命令附加 `--help` 选项，可以显示所有可用于配置该应用程序的操作。

列表 7-28 配置帮助

```
~/nmap 4.76$ ./configure --help
'configure' configures this package to adapt to many kinds of systems.

Usage: ./configure [OPTION]... [VAR=VALUE]...

To assign environment variables (e.g., CC, CFLAGS...), specify them as
VAR=VALUE. See below for descriptions of some of the useful variables.

Defaults for the options are specified in brackets.

Configuration:
-h, --help                display this help and exit
  --help=short            display options specific to this package
  --help=recursive        display the short help of all the included packages
-V, --version             display version information and exit
-q, --quiet, --silent     do not print 'checking...' messages
  --cache-file=FILE       cache test results in FILE [disabled]
-C, --config-cache        alias for '--cache-file=config.cache'
-n, --no-create            do not create output files
  --srcdir=DIR            find the sources in DIR [configure dir or '..']

Installation directories:
  --prefix=PREFIX          install architecture independent files in ➡
PREFIX [/usr/local]
```

■注：列表 7-28 已经详细说明了 `configure` 命令之前的 `./` 的含义。它告诉 Linux 系统去运行在当前目录中找到的 `configure` 脚本。

帮助文档后面的内容还有很多，但是我们将着眼于最重要的一项，即 `--prefix`。这个选项决定软件将安装在什么地方，重要的是这个目录不能被软件包程序使用。否则，你可能会遇

到这种情况：由源代码安装的文件将覆盖软件包文件，这将会使软件包管理器混乱。当移除存在问题的软件包时，被编译文件将被删除。

当由源代码安装应用程序时，它们通常被放置在 `/usr/local` 目录中。这通常是 `--prefix` 的默认选项（可以从列表 7-28 中的方括号中查到这个默认选项）。

其他需要密切注意的是那些决定程序可用功能的选项。这些选项决定第三方库在配置的时候是否应该被检查以及存在时是否被使用。这些选项通常以 `--with-` 和 `--without-` 作为前缀。

掌握了这些新知识以后，我们可以用默认的选项对 `nmap` 源代码进行配置，正如列表 7-29 中所示。

列表 7-29 配置源代码树

```
~/nmap-4.76$ ./configure --prefix=/usr/local
checking build system type... i686-pc-linux-gnu
checking host system type... i686-pc-linux-gnu
checking for gcc... gcc
<snip>
checking openssl/ssl.h presence...-no
checking for openssl/ssl.h... no
configure: WARNING: Failed to find openssl/ssl.h so OpenSSL will not be used. ➡
If it is installed you can try the --with-openssl=DIR argument
<snip>
checking if struct ip has ip_sum member...-yes
configure: creating ./config.status
config.status: creating Makefile
config.status: creating nsock_config.h
<snip>
/ \
NMAP IS A POWERFUL TOOL --USE CAREFULLY AND RESPONSIBLY
Configuration complete. Type make (or gmake on some *BSD machines) to compile.
~/nmap-4.76$
```

配置脚本用来检查系统中编译器的存在、检查所需的头文件，以及头文件中关于函数和数据结构的定义。本章开始部分已经讨论过头文件及库，并且提到只有在由源代码编译程序时才会用到它们。在列表 7-29 中。注意脚本无法找到 OpenSSL 的头文件。所以它会构建不支持 OpenSSL 的 `nmap`。如果你希望得到支持 OpenSSL 的 `nmap`，则需要在配置之前安装适当的 `-devel` 或 `-dev` 软件包。比如，在 Ubuntu 上安装 OpenSSL 的头文件可以使用如下指令。

```
$ sudo aptitude install libssl-dev
```

■注：如果你使用的是基于 Red Hat 的系统，你需要安装 `openssl-devel` 包。

在某些情况下，你仍然可以编译没有头文件的软件。与那些头文件或库相关的功能可能只是被禁用，正如列表 7-29 中所示。但是在其他情况下，配置过程可能会失败。

7.6.2 编译和生成

完成之后，`configure` 命令会生成一个配置头文件和一个称为 `Makefile` 的特殊文件。前者的代码用于编译器对可用的和库进行编译，后者则包含用 `make` 命令组建软件时所需的一些命令。`make` 命令读取 `Makefile` 文件并且执行其中的命令以及包含在里面的步骤。在列表 7-30

里我们执行 `make` 命令来组建 `nmap`。

列表 7-30 编译 `nmap`

```
~/nmap-4.76$ make
Makefile:262: makefile.dep: No such file or directory
g++ -MM -Iliblua -Ilibnet-stripped/include -Ilibpcap -Inbase ➤
-Insock/include main.cc nmap.cc targets.cc tcpip.cc nmap_error.cc utils.cc ➤
idle_scan.cc osscan.cc osscan2.cc output.cc scan_engine.cc timing.cc charpool.cc ➤
services.cc protocols.cc nmap_rpc.cc portlist.cc NmapOps.cc TargetGroup.cc ➤
Target.cc FingerPrintResults.cc service_scan.cc NmapOutputTable.cc MACLookup.cc ➤
nmap_tty.cc nmap_dns.cc traceroute.cc portreasons.cc nse_main.cc nse_nsock.cc ➤
nse_init.cc nse_fs.cc nse_nmaplib.cc nse_debug.cc nse_pcrelib.cc nse_binlib.cc ➤
nse_hash.cc nse_bit.cc > makefile.dep
Compiling liblua
make[1]: Entering directory '/home/jsmith/nmap-4.76/liblua'
gcc -O2 -Wall -g -O2 -Wall -fno-strict-aliasing -DHAVE_CONFIG_H ➤
-DNMAP_NAME="Nmap" -DNMAP_URL="http://nmap.org" ➤
-DNMAP_PLATFORM="i686-pc-linux-gnu" -DNMAPDATADIR="/usr/local/share/nmap" ➤
-DNMAPLIBEXECDIR="/usr/local/libexec/nmap" -DLUA_USE_POSIX ➤
-DLUA_USE_DLOPEN -c -o lapi.o -c lapi.c
<snip>
changing mode of build/scripts-2.5/zenmap from 644 to 755
make[1]: Leaving directory '/home/jsmith/nmap-4.76'
```

如果 `make` 命令成功执行，则生成相关的应用文件。反之，你则会收到一个错误信息，提示相关错误原因以及可行的修改方法。有很多原因可能导致应用程序组建的失败——原因太多不在此赘述——但通常你可能会遇到的是别人以前发现过的问题。你所遇到的问题可能在该应用程序的相关网站上会有详细叙述——比如，在安装文件或在 FAQ 部分。并且，通过搜索特定的错误信息，Google 可以帮你找到可能的解决方法。

7.6.3 安装

现在已经完成 `nmap` 应用程序的编译，为了使系统中的所有用户能够使用该程序，我们只需将它安装在先前配置时设定的 `prefix` 位置。`Makefile` 文件也包括了实现这一目标的命令，其安装过程如列表 7-31 所示。

列表 7-31 安装 `nmap`

```
~/nmap-4.76$ sudo make install
[sudo] password for jsmith:
/usr/bin/install -c -d /usr/local/bin /usr/local/share/man/man1 ➤
/usr/local/share/nmap
/usr/bin/install -c -c -m 755 nmap /usr/local/bin/nmap
/usr/bin/strip -x /usr/local/bin/nmap
/usr/bin/install -c -c -m 644 docs/nmap.1 /usr/local/share/man/man1/
<snip>
ln -sf zenmap /usr/local/bin/xnmap
NMAP SUCCESSFULLY INSTALLED
```

因为普通用户不被允许在 `/usr/local` 下创建新文件，所以需要使用 `sudo` 命令。同上，通过

在 Makefile 文件中写入处理规则并执行这些命令来在系统中安装 nmap 及与其相关的文件。现在我们可以运行新安装的应用程序来确定它是否能够顺利运行，如列表 7-32 所示。

列表 7-32 运行 nmap

```
nmap-4.76$ /usr/local/bin/nmap -V
Nmap version 4.76 ( http://nmap.org )
```

7.6.4 卸载

在管理安装源文件时，从系统内卸载程序将会很棘手。因为并不是全部的程序都包含卸载命令。对于 nmap，我们可以调用下面的这条命令。

```
~/nmap-4.76$ sudo make uninstall
```

然而，这条命令需要系统中相应的配置文件。但不太理想的是，我们不仅需要记录从源文件安装了哪些软件，而且还要记录保存源文件的位置。这就是我们推荐读者通过软件包安装应用程序而避免从源文件安装的一个原因。

7.7 小 结

在这一章，我们已经了解到 Linux 主机如何管理软件，学会了如何管理已安装的软件包，如何添加和卸载它们，以及如何更改软件库来添加更多的应用软件。

你现在应该具备查看主机上安装了哪些软件包以及通过使用各种包管理器来安装和卸载软件包的能力。同时也学会了如何更新主机上已安装的软件包、更改主机上的软件库，以及如何从源文件进行基本的安装。

下一章将学习如何设置存储以获得最大的可靠性，读者将学到如何避免硬盘错误以及如何恢复。

第 8 章 存储管理与灾难恢复



在安装第一台 Linux 主机时，你对磁盘和分区的设置采用了缺省值。既然现在了解了一些基本的系统管理知识，就让我们再回到存储配置中，看看如何修改它以适应需要。我们将考虑各种类型的存储硬件以及使用存储管理软件的有利方式。无论什么事务，数据都是一个关键组成部分，因此要确保它安全、可访问，并始终保持如此。

本章将讲述如何创建和管理磁盘分区与 RAID，如何使应用程序可访问存储器，以及如何从系统故障中恢复。

■注：第 13 章将阐述如何备份和恢复数据。

8.1 存储器基础

在一开始，我们要看看 Linux 是如何处理存储器的。为此，我们要添加各种新磁盘并为它们分区，然后对这些存储器进行格式化和管理的。

Windows 上的驱动器在格式化后显示为驱动器号，而在 Linux 上不是这样。它没有驱动器号的概念，而且格式化的方式也不太一样。不同在于，驱动器和存储器都显示为设备，设备可以分区。而分区可以格式化。

让我们从 Linux 存储器的基本构成块“设备”开始，然后再去讨论分区和文件系统。

8.1.1 设备

第 3 章简要介绍了设备文件。通过这些文件，Linux 使得如硬盘驱动器和 DVD 驱动器之类的硬件设备可从操作系统内部访问。主机里大部分——不是所有的——设备都以文件的形式保存在 `/dev` 目录里。

`/dev` 目录是一个由称作 `udev` 的服务所使用的特殊目录。如果主机启动时内核检测到一个设备，它就告诉 `udev`，然后 `udev` 就在 `/dev` 目录里创建此设备的描述文件。这些设备文件就

是内核提供给应用程序和服务访问设备的方式。

设备文件有好几类，但本章只讨论与存储器有关的类别，这些类别都属于块设备的范畴。块设备涵盖硬盘、磁带驱动器、CD 和 DVD 驱动器，甚至还包括软盘驱动器。所有类型的硬盘——如 ATA、串行 ATA、SCSI 和 SAS——都表示为设备文件，文件名以 sd 开头，代表 SCSI 磁盘，这样访问所有这些不同类型的驱动器就好像访问 SCSI 驱动器一样。

■注：SCSI 是“小型计算机系统接口”（Small Computer System Interface）的首字母缩写词。这是一个有关计算机应如何连接和访问存储设备的规范。要了解更多有关此规范的信息，可以访问 <http://en.wikipedia.org/wiki/SCSI>。

■注：在老的内核版本上或者带有某些 IDE 控制器的系统中，ATA 磁盘可能被检测为 /dev/hd?，而不是 /dev/sd?。不用担心——只是在使用 sd 的地方使用了 hd。

主机上的磁盘设备可以通过使用 ls 命令列出，如列表 8-1 所示。

列表 8-1 设备节点列表

```
$ ls -l /dev/sd*
brw-rw---- 1 root disk 8, 0 2008-12-19 20:24 /dev/sda
brw-rw---- 1 root disk 8, 1 2008-12-19 20:24 /dev/sda1
brw-rw---- 1 root disk 8, 2 2008-12-19 09:26 /dev/sda2
brw-rw---- 1 root disk 8, 5 2008-12-19 20:24 /dev/sda5
```

列表 8-1 显示了 4 个块设备（或设备节点）。它们可由 root 用户和 disk 组读写。接下来通常情况下应该是文件大小，而实际上是两个用逗号隔开的数字。它们是主设备号和次设备号。主设备号告诉内核使用哪个设备驱动器来访问设备，次设备号提供有关此设备的详细信息。最后显示的是此设备文件上次修改的日期和时间。

实际的设备文件名由前缀 sd 和一个表示它所属磁盘的字母组成。第一个被检测到的磁盘是 sda，第二个是 sdb，第三个是 sdc，依次类推。最后，磁盘上每个分区也都有各自的设备节点，分区号是名字的最后部分。这就是说，sda1 是磁盘 sda 的第一个分区，sdb2 是磁盘 sdb 的第二个分区，依次类推。我们很快就将介绍分区。

■注：因为次设备号可为 1~255，并且每个磁盘需要 16 个号码，所以在用完它的设备节点之前，Linux 可以容纳 16 个硬盘，而每个硬盘都是 16 个分区，从 /dev/sda1 到 /dev/sdp16。到 <http://www.linux-tutorial.info/modules.php?name=MContent&pageid=94> 可以找到更多有关设备号及其用途的信息。

如果有一个硬 RAID 控制器，阵列和分区的命名可能就不一样了。RAID 控制器把多个磁盘合并到一个“廉价磁盘冗余阵列”（Redundant Array of Inexpensive Disks，RAID）。本章稍后会详细介绍 RAID。为了找出 RAID 阵列的设备节点，你使用下面的命令可以列出 /dev/ 目录里的所有块设备。

```
$ ls -l /dev | grep ^b
```


这条命令列出的仅仅是 b 打头的行。还是检查内核内部的日志缓存内容会更精确。每当发生一次内核事件，它都会被添加到内核内部的日志缓存里。然后，日志缓存由日志后台程序写入日志文件，使用 `dmesg` 命令可以直接查询它。

```
$ dmesg |less
```

大多数 RAID 控制器还至少使用部分的内核 SCSI 子系统。通过 `less` 里内置的搜索功能，你可寻找已检测到的 SCSI 设备。在 `less` 窗口里输入 `/scsi`，按回车键就可搜索到包含字符串 `scsi` 的所有行。按 `n` 键跳到下一个匹配结果。

8.1.2 分区

在把磁盘添加到主机后，你需要执行几个步骤使它可以使使用。首先，你在磁盘上创建一个或多个分区。磁盘的起始位置必须包含关于分区布局的信息。为了使磁盘可用，即使不打算分割磁盘，你也要创建一个分区。

通常，我们用分区的方式把物理磁盘分成多个可独立使用的虚拟磁盘，就像在 Windows 上把一个磁盘分割成 C:、D: 和 E: 驱动器一样。这样就保持了日志和用户数据与操作系统的隔离，因此日志和用户就不能填满系统磁盘，不会引起问题。

第 5 章提到，引导记录被保存在磁盘的前 224 字节里。描述分区信息的数据则保存在紧挨启动记录的之后 64 字节里。64 字节不能存储很多数据，因此一个磁盘可拥有的分区数目原本相当有限。

物理分区、扩展分区和逻辑分区

分区有三类：物理分区、扩展分区和逻辑分区。这是因为可用的 64 字节只能存储有限的分区信息。描述一个分区需要 16 字节，于是 4 个分区就满了。

作为权变措施，扩展分区概念就被发明出来了。把这 4 个物理分区中的一个标记为扩展分区，它就像一个容器，可以容纳多达 15 个额外的逻辑分区。

描述每个分区的 16 字节包括关于分区的类型（在磁盘上可以找到）和是否为启动盘（尽管 Linux 并不关心这一点）的信息。

要查找更多有关分区的详细信息可访问 <http://www.win.tue.nl/~aeb/partitions/>。

使用 `fdisk` 实用程序可以创建和删除分区。让我们先从已存在的分区开始，列表 8-2 列出了第一个磁盘的分区。因为只有 `root` 用户可读写原始磁盘设备，所以你要使用 `sudo`。

列表 8-2 fdisk 程序显示的分区列表

```
$ sudo fdisk -l /dev/sda

Disk /dev/sda: 4294 MB, 4294967296 bytes
255 heads, 63 sectors/track, 522 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x000c79c5

   Device   Boot    Start    End  Blocks   Id  System
/dev/sda1   *         1     31   248976   83   Linux
/dev/sda2             32    522  3943957+    5   Extended
/dev/sda5             32    522   394392    8e   Linux LVM
```


从列表 8-2 可以看到，安装程序创建了 3 个分区。

- 为操作系统创建的物理分区。
- 扩展分区。
- 为了使用 LVM 创建的逻辑分区。

虽然不想修改系统盘，但是假设购买了一个新的硬盘，你需要给它分区才可以开始使用它存储数据。首先，你需要保证磁盘被操作系统检测到，并查看它的设备名字。当内核启动时，它打印所有检测到的设备信息。登录之后，你可以通过 `dmesg` 命令访问这些信息。

```
$ dmesg | grep sd
[ 9.706404] sd 2:0:0:0: [sda] 8388608 512-byte hardware sectors (4295 MB)
[ 9.706434] sd 2:0:0:0: [sda] Write Protect is off
[ 9.706437] sd 2:0:0:0: [sda] Mode Sense: 5d 00 00 00
[ 9.706486] sd 2:0:0:0: [sda] Cache data unavailable
[ 9.706489] sd 2:0:0:0: [sda] Assuming drive cache: write through
[ 9.706613] sda: sda1 sda2 < sda5 >
[ 9.709163] sd 2:0:0:0: [sda] Attached SCSI disk
[ 9.709540] sd 2:0:1:0: [sdb] 16777216 512-byte hardware sectors (8590 MB)
[ 9.709569] sd 2:0:1:0: [sdb] Write Protect is off
[ 9.709572] sd 2:0:1:0: [sdb] Mode Sense: 5d 00 00 00
[ 9.709617] sd 2:0:1:0: [sdb] Cache data unavailable
[ 9.709620] sd 2:0:1:0: [sdb] Assuming drive cache: write through
[ 9.709684] sdb: unknown partition table
[ 9.710185] sd 2:0:1:0: [sdb] Attached SCSI disk
[ 15.739820] EXT3 FS on sda1, internal journal
```

使用 `grep` 只显示包含 `sd` 的行，这样就把输出限定为有关 SCSI 磁盘子系统的信息。

提示： 了解如何解释启动信息可以帮助解决启动的问题。查找更多有关 `dmesg` 的信息可访问 <http://www.linfo.org/dmesg.html>。

系统检测到两个磁盘：`sda` 和 `sdb`。检测 `sda` 时发现分区 `sda1`、`sda2` 和 `sda5`。`sda5` 使用了尖括号 (`<sda5>`)，说明它是一个逻辑分区。另一个盘是新盘，没有分区表 (`sdb: unknown partition table`)，那么就让我们创建一个。

```
$ sudo fdisk /dev/sdb
```

```
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel with disk identifier 0xf541816d.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.
```

```
The number of cylinders for this disk is set to 1044.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)
```

```
Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)
```

```
Command (m for help): m
```


正如所见，`fdisk` 没有检测到它能识别的任一类型的分区表，并且预置了一个新的 DOS 磁盘标签——在 x86 结构的硬件上 Linux 和 Windows 都可用的分区表类型。

按 `m` 键可看到选项列表。

```
Command (m for help): m
Command action
a  toggle a bootable flag
b  edit bsd disklabel
c  toggle the dos compatibility flag
d  delete a partition
l  list known partition types
m  print this menu
n  add a new partition
o  create a new empty DOS partition table
p  print the partition table
q  quit without saving changes
s  create a new empty Sun disklabel
t  change a partition's system id
u  change display/entry units
v  verify the partition table
w  write table to disk and exit
x  extra functionality (experts only)
```

让我们对这些选项快速浏览一遍。`a` 选项对分区设置可启动的标志，正如在“物理分区、扩展分区和逻辑分区”补充知识中所提到的那样，Linux 忽略此选项。

`b` 选项允许对 BSD 磁盘标签进行编辑。这是 BSD 操作系统所使用的一种备选分区方式。

DOS 兼容性标志可由选项 `c` 打开或关闭。这个标志用来确保超过 1024 个柱面的磁盘在 Linux 下创建的分区能为 Microsoft 的操作系统所用。此选项默认打开，而且实在没理由关掉它。

使用 `d` 选项可以删除分区。通过 `l` 选项可以列出 `fdisk` 能识别的分区类型。此选项显示 16 进制的标识符，可以用在带 `t` 选项的命令中。

创建分区可使用 `n` 选项，它会开启一个向导指导完成创建的全过程，这一点马上就会看到。使用 `o` 选项将清除目前的分区图并创建一个空的新分区图。这比逐个删除所有分区更快，能节省很多时间。

按 `p` 键可列出当前的分区表。它列出的是存在于内存中而不是磁盘中的分区表。

如果不想保存已做的改动，请按 `q` 键，这将退出 `fdisk` 而不会把修改过的分区表写入磁盘。当在 Sun 公司的 SPARC 系统上使用 Linux 时，需要一种不同的分区方法，这可以通过 `s` 选项进行。

分区还保存它们所包含文件系统的类型信息。从 `l` 选项得到的 16 进制标识符可通过 `t` 选项设置。

如果想以扇区而非柱面为单位显示分区大小，请按 `u` 键。`v` 选项会检查分区图是否正确并显示未分区空间的信息。

如果对新的分区图满意，可按 `w` 键把它保存到磁盘。最后，`x` 选项可访问 `fdisk` 高级选项，如磁盘配置的修改、分区内数据的移动等。我们不涉及这些高级选项的使用。

现在按 `p` 键打印当前磁盘上的分区列表，你可以发现它是空的。我们通常推荐在一个数

据磁盘上只创建一个单独分区，但是让我们玩一玩这个磁盘，多创建几个：1 个主分区和 2 个逻辑分区。

先创建 1 个大小为 2GB 的主分区。柱面号可以忽略。按回车键，对第一个柱面采用默认值。这将把此分区置于磁盘的开头。接下来，输入+2G 指明要创建的分区大小。

```
Command (m for help): n
Command action
  e extended
  p primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-1044, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-1044, default 1044): +2G
```

接下来添加一个扩展分区，它占据磁盘的整个剩余部分。设置此分区的编号为 4，以留有空闲分区号。如果将来某个时候想把分区 1 分成多个小的分区，这些空闲号就用得着了。当被问到最后一个柱面时，还是按回车键接受默认值。

```
Command (m for help): n
Command action
  e extended
  p primary partition (1-4)
e
Partition number (1-4): 4
First cylinder (245-1044, default 245): hit enter
Using default value 245
Last cylinder or +size or +sizeM or +sizeK (245-1044, default 1044):
Using default value 1044
```

既然有了一个扩展分区，就可以在它内部创建逻辑分区。让我们从一个 4GB 的分区开始。与主分区或扩展分区相反，这些逻辑分区会被自动编号。

```
Command (m for help): n
Command action
  l logical (5 or over)
  p primary partition (1-4)
l
First cylinder (245-1044, default 245):
Using default value 245
Last cylinder or +size or +sizeM or +sizeK (245-1044, default 1044): +4G
```

最后，接受对最后一个柱面的默认值，使扩展分区的剩余部分可用。

```
Command (m for help): n
Command action
  l logical (5 or over)
  p primary partition (1-4)
l
First cylinder (732-1044, default 732):
Using default value 732
Last cylinder or +size or +sizeM or +sizeK (732-1044, default 1044):
Using default value 1044
```


现在让我们看看已做的事情，打印分区列表。

```
Command (m for help): p

Disk /dev/sdb: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x5ecec006
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sdb1		1	244	1959898+	83	Linux
/dev/sdb4		245	1044	6426000	5	Extended
/dev/sdb5		245	731	3911796	83	Linux
/dev/sdb6		732	1044	2514141	83	Linux

现在有 4 个分区。从它们的开始和结束块的号码可以看出，分区 5 和 6 包含于分区 4 的空间内。

默认情况下，fdisk 所创建的分区 ID（即分区类型）是 Linux。Linux 本身一般不在意分区类型，但为了更易于管理，我们推荐对分区类型进行修改以使之符合预期的用途。我们希望把/dev/sdb6 用作交换分区，因此要修改它的类型。

键入 t 以改变类型，然后选择分区号码 6。

```
Command (m for help): t
Partition number (1-6): 6
Hex code (type L to list codes): L
```

这里将呈现一长串可供选择的分区类型。分区 ID 由两个 16 进制数字组成，不过幸运的是，按 L 键可以得到分区类型列表以及对每种类型的描述。我们正想要 Linux 交换区，因此应该选择的 ID 为 82。表 8-1 给出了最常选择的分区类型。类型 c 很可能在 Windows、Mac OS 和 Linux 系统都会用的 USB 便携存储设备上遇到。

表 8-1 Linux 常用的分区类型

ID	描述
82	Linux 交换区/Solaris
83	Linux
da	Non-FS 数据（用于软 RAID）
fd	Linux RAID 自动检测（以前用于软 RAID）
8e	Linux LVM
c	W95 FAT32（LBA）
7	HPFS/NTFS

输入 82 选择 Linux 交换区类型。

```
Hex code (type L to list codes): 82
Changed system type of partition 6 to 82 (Linux swap / Solaris)

Command (m for help): p

Disk /dev/sdb: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
```



```
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x8b1c93f9
```

```
/dev/sdb1    1      244    1959898+   83   Linux
/dev/sdb4   245     1044    6426000    5   Extended
/dev/sdb5   245      731    3911796    83   Linux
/dev/sdb6   732     1044    2514141    82   Linux swap / Solaris
```

此时，新的分区图只存在于 RAM 中，因为还没有把这些改动保存到磁盘。按 w 键保存并退出 fdisk。如果不想保存这些改动，可按 q 键退出 fdisk。

```
Command (m for help): w
The partition table has been altered!

Calling ioctl() to reread partition table.
Syncing disks.
```

内核重新载入分区图并为分区创建新的设备节点。在 dmesg 的输出中，可以看到磁盘检测程序已经运行并发现了新分区。你也可以验证它们的设备节点已存在于磁盘中。

```
$ ls -l /dev/sdb*
brw-rw---- 1 root disk 8, 16 2008-11-28 16:32 /dev/sdb
brw-rw---- 1 root disk 8, 17 2008-11-28 16:32 /dev/sdb1
brw-rw---- 1 root disk 8, 18 2008-11-28 16:32 /dev/sdb4
brw-rw---- 1 root disk 8, 19 2008-11-28 16:32 /dev/sdb5
brw-rw---- 1 root disk 8, 20 2008-11-28 16:32 /dev/sdb6
```

有时内核不能重读分区表，这意味着在重启主机之前不能访问新分区的设备文件。如果正在编辑的磁盘的某个分区仍然挂载于系统，就会出现这种情况。为避免重启，我们要确保正在进行分区的磁盘没有分区被挂载。本章稍后将讲述分区的挂载。

■注：你也可以通过运行 partprobe 命令使内核重新检测分区而不用重启。

另外一个可创建和删除分区的实用程序是 parted。与 fdisk 不同，它允许编辑分区的大小和次序。我们推荐对分区调整大小不要使用 parted，而要使用 LVM。本章后面将详细介绍 LVM。有关 parted 的更多信息，请访问 <http://www.gnu.org/software/parted/index.shtml>。

■注：对分区调整大小可能导致不可恢复的数据丢失。始终要先备份数据！

补充知识：GIBIBYTES 与 GIGABYTES

当硬盘制造商广告它的产品时，它希望可用存储空间看起来越大越好，因此它使用这样的换算方法：每吉字节等于 1000 兆字节，每兆字节等于 1000 千字节，而每千字节等于 1000 字节。然而，计算机上所有的计算都是按照二进制算术进行的，因此真实的换算因数为 1024。可是如果存储器制造商使用这个换算因数，它的设备与竞争者相比会更小，因此它不会这么做。为了避免换算方式的混乱，人们为换算因数为 1024 的单位设计了新名词：kibibyte、mebibyte、gibibyte，等等。它们表示为 KB、MB、GB，等等。Linux 文件系统使用 1024 这个换算因数，因此如果购买了一个 500GB 的磁盘，通过 Linux 看它的大小总是小于 500GB。更多相关信息请参阅 <http://en.wikipedia.org/wiki/Gigabyte>。

8.1.3 文件系统

现在分区已经创建，但还没有做好使用它的准备。接下来需要做的事情是创建一个文件系统。在 Microsoft Windows 领域里，这被称作格式化。

文件系统有点像图书馆。它存储大量的数据，并且有一个目录来确保人们能找到他们要找的东西。通道与书架的布局以及目录的设计，决定查找和检索任意一条特定信息所花费的时间。创建文件系统就像预置目录并把书架搬进在其他方面还空白的图书馆。

正如没有对所有图书馆都最优的通道和书架布局一样，也不存在对所有应用都“最好”的文件系统。我们不会深入到大量的细节里，但是最常用的 Linux 文件系统可以看一看。表 8-2 列出了这些文件系统及其主要特征。

表 8-2 Linux 文件系统及其主要特征	
文件系统	特征
EXT2	稳定、通用、可缩减或扩充
EXT3	稳定、通用、快速恢复、可缩减或扩充
EXT4	新的、通用、快速恢复、ext3 的改进版
XFS	稳定、通用、快速恢复、可在线扩充
JFS	稳定、通用、快速恢复

ext2 和 ext3 文件系统一般在存储很多小文件时表现较好。对于电子邮件、网站或者办公文件的存储，二者是不错的选择，因为这些应用通常都是由很多大小为几百 K 字节的文件组成。

ext4 是吸取 ext3 教训后开发的一个改进版。此文件系统最近才变得稳定。它提供一些 ext3 没有的特征，如在线碎片消除、更高的日志可靠性和更快的文件系统检查。ext4 的目标是成为有极好性能的综合性文件系统。

如果需要存储视频、大的图像或者数据库文件，那么 XFS 文件系统是一个不错的选择，因为它在处理大到几 GB 的文件方面比 ext3 表现更好。它有某些与 ext4 一样的优点，但因为 XFS 已经稳定了较长时间，也许它是服务器系统更好的选择。

最后，JFS 被认为是一个不错的综合性日志式文件系统，它在处理各种容量和大小的文件方面都做得不错。它还被认为是一种轻型的文件系统，因为它在繁重的磁盘活动期间不会使用大量的 CPU 资源。

要查找详尽的文件系统列表以及它们的特征对比，可以访问 http://en.wikipedia.org/wiki/List_of_file_systems 和 http://en.wikipedia.org/wiki/Comparison_of_file_systems。

传统上，Linux 使用 ext2 (second extended file system, 第二次扩展的文件系统) 文件系统来存储数据，而最近以来，ext3 变成了默认的选项。假设出现系统故障，ext2 必须先扫描整个文件系统才能被再次挂载和使用。当文件系统只有 2GB，这没问题，但是对于今天的磁盘，文件系统可以有几 TB 的容量，检查它们将花费数小时甚至数天，在此期间文件系统的数据将不可用。

为了解决这种问题，人们创建了日志式文件系统。ext3 和 ext4 都是这样的文件系统，从而没有 ext2 中存在的那种很长的恢复等待时间。更多信息请看补充知识“日志式文件系统”。

补充知识：日志式文件系统

想象在图书馆里，为了一本归还的书，图书管理员从前台走开，找到某处书架上的空位，把书放进去，更新目录，然后回到前台处理下一本归还的书。一直这样的话，不可能有任何人借到一本书。

有了还书槽，这个问题就可以解决了。图书管理员可以在图书馆不忙于为读者借新书时处理归还的书。即使还书槽中的书在图书馆关闭之前没被处理，它们也不会丢。第二天它们还在那儿。

日志式文件系统有几分像带还书槽的图书馆。任何需要写入磁盘的信息都先放在日志里，稍后在操作系统有空闲时间时再把它放入它在磁盘的最终位置上。类似地，即使机器崩溃了，日志中的数据也不会丢。在下次挂载文件系统时日志被简单处理或者重放。

不过，这个比喻到此为止。在文件系统这个图书馆里，人们还可以从还书槽借书，如果还书槽变得太满，图书管理员可以忽略要借书的人。

大多数现代文件系统使用日志，尽管某些文件系统只对文件元数据使用日志。为了得到额外的速度，一些文件系统允许把日志存储于单独的设备——如固态硬盘（solid-state drive, SSD）。关于 SSD 设备的更多信息可访问 http://en.wikipedia.org/wiki/Flash_drive。

1. 创建文件系统

早前创建的最后一个分区/dev/sdb6 较小，将被用作交换分区。给这种分区选择文件系统很简单，因为只有一种交换区文件系统格式。我们先使用 mkswap 命令设置它，如列表 8-3 所示。

列表 8-3 设置交换区

```
$ sudo mkswap /dev/sdb6
Setting up swapspace version 1, size = 2574475 kB
no label, UUID=96570203-0efd-4c18-8887-217d873a6051
```

mkswap 程序把/dev/sdb6 标记为交换区。你可以利用生成的 UUID 添加一个条目到 /etc/fstab 文件，此文件列出了主机上用到的所有文件系统（请看补充知识“UUID”以了解 UUID）。本章后面将介绍/etc/fstab 文件。从技术上讲，这不是对分区格式化，而是写入少量信息，向内核表明它是作交换区用的。

通过 swapon 命令你可立即激活新的交换分区。此命令告诉内核它可以用指定的分区作为交换区。

```
$ sudo swapon /dev/sdb6
```

此命令执行完不打印任何结果，但你可以通过检查 dmesg 来了解。把输出结果放入 tail，以限制行数为指定的数字。

```
$ dmesg | tail -n 1
[35980.499562] Adding 262136k swap on /dev/sdb6.
Priority:-1 extents:1 across:262136k
```

还有一种检查交换区的方式，就是看 free 命令的报告。指定-m 选项使显示大小的单位为 MB。


```
$ free -m
      total        used        free      shared    buffers     cached
Mem:      503          190         313           0          61          69
-/+ buffers/cache:      58         444
Swap:      255           0        255
```

此命令报告交换区总共为 255MB，这的确是刚添加的交换区大小。在第 17 章介绍性能管理时会回到 free 命令。

补充知识: UUID

在 Windows 上安装软件时或者在某些网站的网址里，你可能已经看到了类似“8d3564ba-92b0-4238-a56f-1c2a0ae85eb2”这样的看似半随机又很长的 16 进制字符串。这些字符串就是通用唯一标识符（Universally Unique Identifier, UUID）。

UUID 为识别信息提供了一种便利的、计算上容易实现的方式，它不需要检查生成的 ID 是否已在使用。因为 UUID 随机或者半随机生成，很难被猜到，所以它也提供了一些安全性。

UUID 作为区分 RAID 阵列、逻辑卷和文件系统等构件的方法，正被越来越多地用在 Linux 上。在 http://en.wikipedia.org/wiki/Universally_Unique_Identifier 上可以了解更多相关信息。

对于数据分区，先从新的 2GB 的/dev/sdb1 开始。我们用 mkfs.ext3 实用程序把它格式化为 ext3 格式，如列表 8-4 所示。除了日志选项，ext3 与 ext2 没什么区别。如果要创建 ext2 格式的文件系统，只需改为运行 mkfs.ext2。

列表 8-4 创建 ext3 文件系统

```
$ sudo mkfs.ext3 /dev/sdb1
mke2fs 1.40.8 (13-Mar-2008)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
122880 inodes, 489974 blocks
24498 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=503316480
15 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912

Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 25 mounts or
180 days, whichever comes first. Use tune2fs -c or -i to override.
```

当此命令执行完，花一些时间看看它的输出。你可以看到此文件系统的标签没有定义。其实使用-L 参数可以定义一个标签。有了这种标签，你就可以用标签名而不是设备名来指代分区。例如，你可以基于设备的用途为它们加上标签。

接着看到的是一系列有关文件系统大小和存储空间怎么分配的统计数字。补充知识“块

与索引节点”对此有简短的解释。比较著名的块是为超级用户和超级块备份所保留的块。

```
24498 blocks (5.00%) reserved for the super user
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912
```

超级块是文件系统元数据的组成成分。它包含文件系统的相关信息，如它的大小、空闲空间容量、数据在文件系统内的位置等。如果发生了系统故障，而且超级块受到了损坏，那就没有办法确定数据在文件系统中的位置。为帮助应付这类事情，超级块在众所周知的块号上保留了好几个备份。本章稍后将重回有关恢复的话题。

超级用户的保留块在文件系统中占有一定的比例，这是为了使普通用户不要使用文件系统太过度，以至于超级用户（root 用户）都不能登入，或者以 root 用户身份运行的服务不能往磁盘写数据。

5%这个限制比例是有根据的、合适的，例如，对于根文件系统，其大小通常不超过几 GB。然而，当文件系统容量有 1TB 时，5%就相等于 50GiB，这么大的空间都不能用来存储用户数据，因此在数据存储卷上修改或者去除这个比例还是有意义的。

你可以在创建文件系统时通过在 `mkfs.ext3` 程序中指定 `-m 0` 选项，把保留块的比例设置为 0，或者以后再更改。

最后，`mkfs.ext3` 表示可以修改文件系统自动检查的间隔时间。默认情况下，自动检查会发生在挂载次数达到 25（文件系统被挂载 25 次）时，或者挂载后 180 天还没有通过检查时。这种行为是不希望发生的，因为这样一次检查会花费数小时。

补充知识：块与索引节点

在创建文件系统时，磁盘空间被分成很多等份。每一等份称为一“块”，大小默认为 4KB。一个块只能放一个文件或者部分文件，因此一个 1KB 的文件占用一个整块，即 4KB 的空间，浪费了 3KB 的存储空间。大些的文件需要分好多块存储。如果主要存储小于 4KB 的文件，你可以选择使用不同的块尺寸。

“索引节点”被大多数文件系统用来存储元数据，如文件或目录的创建日期、修改日期、权限、所有者，以及块所包含实际文件数据的指针。这意味着，文件系统只能包含和它的索引节点一样多的文件和目录。那么如果块尺寸极小而文件很多，就可能在用完磁盘空间之前先用光了索引节点。JFS 没有这个限制，因为在需要时它会自动增加额外的索引节点。要了解更多有关索引节点的信息，请参阅 <http://en.wikipedia.org/wiki/Inode>。

2. 调整 ext2、ext3 和 ex4 文件系统选项

`ext2`、`ext3` 和 `ext4` 文件系统创建以后，可使用 `tune2fs` 实用程序修改其选项。为了总览一下各选项，首先不带任何参数地运行程序。你也可以通过命令 `man tune2fs` 显示整个手册。

```
$ tune2fs
tune2fs 1.40.8 (13-Mar-2008)
Usage: tune2fs [-c max_mounts_count] [-e errors_behavior] [-g group]
    [-I interval[d|m|w]] [-j] [-Jjournal_options] [-l]
    [-m reserved_blocks_percent] [-o [^]mount_options[,...]]
    [-r reserved_blocks_count] [-u user] [-C mount_count] [-L volume_label]
    [-M last_mounted_dir] [-O [^]feature[,...]]
```



```
[-E extended-option[,...]] [-T last_check_time] [-U UUID] device
```

虽然没有明确说，实际上，`-l` 参数可以列出当前文件系统的选项。让我们在新的 `ext3` 分区上运行此命令（见列表 8-5）。

列表 8-5 显示 `ext2`、`ext3` 或者 `ext4` 的文件系统选项

```
$ sudo tune2fs -l /dev/sdb1
tune2fs 1.40.8 (13-Mar-2008)
Filesystem volume name:      <none>
Last mounted on:             <not available>
Filesystem UUID:             f06da31f-ac2e-4e66-9f71-81373a36086e
Filesystem magic number:     0xEF53
Filesystem revision #:       1 (dynamic)
Filesystem features:         has_journal ext_attr resize_inode dir_index filetype
                             sparse_super large_file
Filesystem flags:             signed_directory_hash
Default mount options:       (none)
Filesystem state:            clean
Errors behavior:             Continue
Filesystem OS type:          Linux
Inode count:                 122880
Block count:                 489974
Reserved block count:        24498
Free blocks:                 477170
Free inodes:                 122869
First block:                 0
Block size:                  4096
Fragment size:               4096
Reserved GDT blocks:         119
Blocks per group:            32768
Fragments per group:        32768
Inodes per group:            8192
Inode blocks per group:      256
Filesystem created:          Thu Dec 18 16:24:16 2008
Last mount time:             n/a
Last write time:             Thu Dec 18 16:24:16 2008
Mount count:                 0
Maximum mount count:         27
Last checked:                Thu Dec 18 16:24:16 2008
Check interval:              15552000 (6 months)
Next check after:            Tue Jun 16 15:24:16 2009
Reserved blocks uid:         0 (user root)
Reserved blocks gid:         0 (group root)
First inode:                 11
Inode size:                  128
Journal inode:               8
Default directory hash:      tea
Directory Hash Seed:         18cbe84b-a400-4677-ab39-76659257aed9
Journal backup:              inode blocks
```

这里显示了很多信息，但我们最感兴趣的是文件系统的 `UUID` 和状态。这里还列出了挂载次数（文件系统被挂载的次数）和检查间隔时间（尽管时间的显示以秒为单位，而不是创建文件系统时的以天为单位）。

■注：在第 17 章阐述容量规划与性能时，我们将详细介绍这些文件系统的某些功能。

首先把挂载间隔和最大挂载次数设置为 0。如果再想做文件系统检查，你可以在启动时手动执行，但这样的话，就不会有本来想快速重启却突然需要数小时等待的意外了。同样地，把保留块百分比设为 0，因为此分区不需要保留空间。

```
$ sudo tune2fs -c 0 -i 0 -m 0 /dev/sdb1
tune2fs 1.40.8 (13-Mar-2008)
Setting maximal mount count to -1
Setting interval between checks to 0 seconds
Setting reserved blocks percentage to 0% (0 blocks)
```

表 8-3 列出了 tune2fs 中最可能使用的选项。

表 8-3 tune2fs 常用选项	
选项	功能
-c N	设置文件系统强制检查前的挂载次数为 N
-l	列出当前文件系统选项
-m N	设置保留块占总块数的比例为 N%
-r N	设置保留块的数量为 N
-j	在文件系统中创建一个日志（把 ext2 变为 ext3）
-L label	给文件系统分配一个标签“label”
-O feat	切换文件系统 feat 功能的开关状态

■注：在第 17 章阐述性能与容量规划时，我们将回到 -O 选项以及文件系统高级功能。

3. XFS 和 JFS 文件系统

XFS 和 JFS 文件系统最初都是有知识产权、闭源的产品。XFS 是 Silicon 公司为它的 IRIX 操作系统所开发的文件系统，而 JFS 是 IBM 为了用于 OS/2 Warp 服务器而开发的文件系统。

两家公司在数年前把他们的文件系统开发软件开源，并努力把它们集成到 Linux 内核中，因为当时 Linux 缺少一个日志式文件系统。业界热情拥抱这些新的开源文件系统，因为两者提供了新的功能和优秀的性能。现在它们都在 Linux 平台上得到了广泛接受与支持。

● XFS

你已经创建了一个 ext3 分区来存储一些小文件。现在让我们用 XFS 文件系统格式化另一个分区，这样就可以有效地存储大文件。为此，你需要使用 mkfs.xfs 工具。因为第一次安装主机时没有创建 XFS 分区，所以就没有安装 XFS 实用工具。此工具由 xfsprogs 软件包提供，你必须安装它。在 Ubuntu 上安装命令如下所示。

```
$ sudo aptitude install xfsprogs
```

在 Red Hat 上的安装命令如下所示。

```
$ sudo yum install xfsprogs
```


安装完成后，你可以使用工具的默认选项创建文件系统，如列表 8-6 所示。

列表 8-6 创建 XFS 文件系统

```
$ sudo mkfs.xfs /dev/sdb5
meta-data=/dev/sdb5          isize=256    agcount=8, agsize=122243 blks
      =                       sectsz=512    attr=0
data      =                   bsize=4096    blocks=977944, imaxpct=25
      =                       sunit=0      swidth=0 blks, unwritten=1
naming    =version 2         bsize=4096
log        =internal log     bsize=4096    blocks=2560, version=1
      =                       sectsz=512    sunit=0 blks, lazy-count=0
realtime  =none              extsz=4096    blocks=0, rtextents=0
```

随着此文件系统的创建，它的一些配置信息也显示出来了。我们在第 17 章讨论性能和容量规划时将用到这些信息。

所有这些选项，如控制块尺寸和日志大小的选项，都可以在创建文件系统时设置，但是 `mkfs.xfs` 工具会基于需要格式化的分区大小选择缺省值。

■注：XFS 不保留总空间的 5%给 root 用户，也不自动做定期的文件系统检查。

● JFS

为了试用 JFS，用 JFS 代替 XFS 重新对 `/dev/sdb5` 格式化。与使用 XFS 时的情况一样，除非初始安装主机时创建了 JFS 分区，否则 JFS 实用工具不会默认安装。这些实用工具由软件包 `jfsutils` 提供。在 Ubuntu 上安装它们的命令如下所示。

```
$ sudo aptitude install jfsutils
```

在 Red Hat 上的命令如下所示。

```
$ sudo yum install jfsutils
```

一旦安装完这些工具，你就要对希望格式化的分区调用 `mkfs.jfs`，如列表 8-7 所示。

列表 8-7 创建 JFS 文件系统

```
$ sudo mkfs.jfs /dev/sdb5
mkfs.jfs version 1.1.11, 05-Jun-2006
Warning! All data on device /dev/sdb5 will be lost!

Continue? (Y/N) y
|
Format completed successfully.

3911796 kilobytes total disk space.
```

在同意销毁目标分区所有数据之后，文件系统就被创建了。与使用 XFS 时的情况一样，JFS 没有保留空间，也没有强制性系统检查的时间间隔。

4. 文件系统与数据共享

到目前为止，已经介绍的文件系统只能由 Linux 访问。如果要在不同的操作系统间传输

数据——例如，把数据传给膝上型电脑或者从客户机往外传送数据——可能会使用到某种文件系统，不仅 Linux 可以访问，Windows 和 Mac OS X 也可以访问。

这方面的事实标准是 FAT 文件系统，它是 Microsoft 为 MS-DOS 开发的文件系统。FAT 有好几种。最新的版本是 FAT32，它支持的磁盘容量超过 32GB，文件大小可达 4GB。

为创建 FAT32 文件系统，你要使用 `mkfs.vfat` 工具。这个工具在 Ubuntu 和 Red Hat 上都由软件包 `dosfstools` 提供，因此要确保已安装此包。

插入要格式化的 USB 便携存储设备后，通过内核记录检查它的设备节点名，如列表 8-8 所示。

列表 8-8 确定 USB 便携存储设备的设备节点

```
$ dmesg
[ 52.464662] usb 1-1: new high speed USB device using ehci_hcd and address 2
[ 52.887506] usb 1-1: configuration #1 chosen from 1 choice
[ 52.967324] usbcore: registered new interface driver libusual
[ 52.981452] Initializing USB Mass Storage driver...
[ 52.986046] scsi3 : SCSI emulation for USB Mass Storage devices
[ 52.987804] usbcore: registered new interface driver usb-storage
[ 52.987831] USB Mass Storage support registered.
[ 52.988661] usb-storage: device found at 2
[ 52.988687] usb-storage: waiting for device to settle before scanning
[ 58.982976] usb-storage: device scan complete
[ 59.350262] usb 1-1: reset high speed USB device using ehci_hcd and address 2
[ 59.772402] scsi 3:0:0:0: Direct-Access      SanDisk Cruzer
8.01 PQ: 0 ANSI: 0 CCS
[ 59.789834] sd 3:0:0:0: [sdg] 15682559 512-byte hardware sectors (8029 MB)
[ 59.792747] sd 3:0:0:0: [sdg] Write Protect is off
[ 59.792754] sd 3:0:0:0: [sdg] Mode Sense: 45 00 00 08
[ 59.792766] sd 3:0:0:0: [sdg] Assuming drive cache: write through
[ 59.805772] sd 3:0:0:0: [sdg] 15682559 512-byte hardware sectors (8029 MB)
[ 59.815884] sd 3:0:0:0: [sdg] Write Protect is off
[ 59.815891] sd 3:0:0:0: [sdg] Mode Sense: 45 00 00 08
[ 59.815894] sd 3:0:0:0: [sdg] Assuming drive cache: write through
[ 59.816480] sdg: sdg1
[ 59.831448] sd 3:0:0:0: [sdg] Attached SCSI removable disk
[ 59.831942] sd 3:0:0:0: Attached scsi generic sg7 type 0
```

在列表 8-8 中，ScanDisk Cruzer USB 驱动器被检测为设备 `/dev/sdg`。一旦知道 USB 驱动器的设备节点，你就可以创建类型为 `c-W95 FAT32 (LBA)` 的主分区，然后使用 `mkfs.vfat` 将它格式化。使用 `-n` 选项给此分区加标签，并通过 `-F 32` 选项表明想要 FAT32 类型的文件系统。

```
$ sudo mkfs.vfat -n "USB Key" -F 32 /dev/sdg1
mkfs.vfat 2.11 (12 Mar 2005)
```

5. 其他文件系统

Linux 上可用的文件系统数量众多，那么你可能问为什么只介绍其中的 3 个。尽管存在很多其他的文件系统，但我们觉得它们大部分不适合或者没有准备好用在产品平台中。文件系统必须具有的最重要特征是稳定性。已介绍的文件系统具有这个特征并且提供了优越的性

能。如果根据要存储的数据类型选择了 ext3、XFS 或者 JFS，那么你应该见识了它优越的可靠性和速度。如果为服务器选择更快但更不稳定的文件系统，每月都要花时间从备份里恢复一次数据，那么这样的选择将是无益的。

为了对 Linux 内核所支持的其他文件系统有一个简要概括的了解，可以阅读 `filesystems` 的手册页面。

■注：通过软件包 `ntfsprogs` 里的 `mkntfs` 工具，Linux 可以创建 NTFS 文件系统。不过，我们不推荐在 Linux 下使用 NTFS 文件系统存储数据。在第 12 章介绍文件共享时会说明如何访问 NTFS 格式的分区。

8.2 使用文件系统

现在你已经在新磁盘 `/dev/sdb` 上创建了分区，并按所选择的文件系统对它们进行了格式化。然而，在使用文件系统存储数据之前，还需要挂载它。

正如第 3 章以及本章开头简要介绍过的那样，Linux 上的文件系统没有分配驱动器号。作为替代，它们以目录的形式挂载在根文件系统或者子目录下某处。在第 3 章提到，`/mnt` 目录常被用来临时挂载文件系统。接下来你要创建一个目录 `/mnt/data`，用于新的 ext3 分区。

```
$ sudo mkdir /mnt/data
```

挂载分区是通过 `mount` 命令完成的。使用 `-t` 选项指定文件系统类型，然后是设备文件，接下来是希望文件系统使用的目录。

```
$ sudo mount -t ext3 /dev/sdb1 /mnt/data
```

如果一切顺利，`mount` 命令不会打印任何信息，只是简单地退出。为了验证分区已被挂载，使用 `df` 命令。

```
$ df
Filesystem      1K-blocks      Used Available Use% Mounted on
/dev/mapper/au--mel--ubuntu--1-root
                3681680      638168    2857964    19% /
varrun          257724         72     257652     1% /var/run
varlock         257724          0     257724     0% /var/lock
udev            257724        112     257612     1% /dev
devshm          257724          0     257724     0% /dev/shm
/dev/sda1       241116        25626    203042    12% /boot
/dev/sdb1       2080592       68680   1907056     4% /mnt/data
```

我们的分区列在输出的底部，因此 `mount` 命令已成功执行。本章稍后将重访 `df` 命令并详细解释其输出的含义。

你还可以通过 `dmesg` 命令检查内核记录，以此核对详细信息。

```
$ dmesg
[692927.431902] kjournald starting. Commit interval 5 seconds
```



```
[692927.440571] EXT3 FS on sdb1, internal journal
[692927.440583] EXT3-fs: mounted filesystem with ordered data mode.
```

内核检测到一个带内部日志的 **ext3** 文件系统并挂载了它。它还启动了一个内核线程，每 **5sec** 刷新一次从日志到文件系统的数据——就像图书管理员清空还书槽。

如果把工作目录改到新挂载的分区，你就可以使用 **ls** 命令看它是否包含什么东西。

```
$ cd /mnt/data
$ ls -l
total 16
drwx----- 2 root root 16384 2008-12-18 16:24 lost+found
```

全新的文件系统包含一个单独的目录 **lost+found**，而我们并没有创建它！这实际上是一个存在于所有 **ext2** 和 **ext3** 文件系统里的特殊目录——在根目录下也可以看到一个这样的目录 **/lost+found**。此目录是给 Linux 文件修复工具使用的，在稍后的“故障恢复”一节会讲到它。

当不再需要此文件系统时，你可以使用 **umount** 命令把它从主机中卸载。

```
$ sudo umount /mnt/data
umount: /mnt/data: device is busy
umount: /mnt/data: device is busy
$ pwd
/mnt/data
```

umount 拒绝卸载此目录，因为它包含正被使用的文件或目录。在本例中，正因为当前工作目录是 **/mnt/data**，所以主机不能在处于该目录时卸载此设备。设备忙的原因很多，而且哪个用户或应用程序打开了哪些文件或目录并不总是清楚的。为了帮忙弄清楚，**lsof** 命令列出了打开的文件和目录。

```
$ lsof /mnt/data
COMMAND PID USER   FD  TYPE  DEVICE  SIZE      NODE     NAME
bash      20932 jsmith cwd   DIR    8,17    4096         2 /mnt/data
lsof      21352 jsmith cwd   DIR    8,17    4096         2 /mnt/data
lsof      21353 jsmith cwd   DIR    8,17    4096         2 /mnt/data
```

除了 **lsof** 本身，还有一个 **jsmith** 用户所有的 **bash** 进程。通过返回主目录可以让此进程停止使用该目录。键入 **cd** 和 **~** 快捷键回到主目录，然后再使用 **lsof** 检查 **/mnt/data**。

```
$ cd ~
$ lsof /mnt/data
```

这次 **lsof** 命令没有返回任何打开的文件和目录。既然该目录不再被列入使用名单中，那么现在就可以安全地卸载它。

```
$ sudo umount /mnt/data
```

■注：妥善地卸载文件系统会把 **tune2fs** 输出中所看到的 Filesystem state 标志设置为“clean”，因为它会要求内核处理整个日志文件并确保所有数据都写入磁盘。这会阻止下次启动主机时文件系统的自动检查。

当作为非 **root** 用户运行 **lsof** 命令时，它只会列出该用户所拥有的进程。正在试图卸载的文件系统上的文件和目录可能仍在被别人使用。如果 **umount** 一直报告文件系统忙碌，那么使用 **sudo** 运行 **lsof** 命令来检查。

■注：如果已挂载的文件系统正被某个系统服务使用，那么必须先停止此服务，然后再卸载文件系统。

8.2.1 自动挂载

你可能注意到其他分区不需要手动挂载。它们在开机时就已经被挂载了。这是启动进程的一部分工作。我们希望在启动时自动挂载的每个分区都要列在/etc/fstab 文件中。列表 8-9 显示了 Ubuntu 主机上的 fstab 文件。

列表 8-9 一个 fstab 文件

```
# /etc/fstab: static file system information.
# <file system> <mount point> <type> <options> <dump> <pass>
proc /proc proc defaults 0 0
# /dev/mapper/au--mel--ubuntu--1-root
UUID=e9e48791-dd89-4bf7-921e-f6d460f7ca24 / ext3 ➡
    relatime, errors=remount-ro 0 1
# /dev/sda1
UUID=74bce18f-6f8f-4f4c-be91-4ce38e2bff8e /boot ext3 ➡
    relatime 0 2
# /dev/mapper/au--mel--ubuntu--1-swap_1
UUID=83f01bff-2cc4-4f4a-abel-0b9210c5dfa8 none swap ➡
    sw 0 0
/dev/scd0 /media/cdrom0 udf,iso9660 user,noauto,exec,utf8 0 0
/dev/fd0 /media/floppy0 auto rw,user,noauto,exec,utf8 0 0
# /dev/sdb6
UUID=96570203-0efd-4c18-8887-217d873a6051 none swap ➡
    sw 0 0
```

文件中每一行都包含 6 个由空格或制表符隔开的字段。它们指明了每个文件系统如何挂载、在哪儿挂载，以及执行检查时做什么。以#号开始的行都是注释。

文件系统字段（file system）包含待挂载文件系统的设备节点名。你也可以通过定义 LABEL=label 或者如例中那样用文件系统的 UUID 代替文件系统标签。Red Hat 默认使用设备节点或标签，而 Ubuntu 默认使用 UUID 号，但它们都支持使用任何一种方式。我们将使用 UUID 号，因为即便磁盘因检测顺序不同而有不同的名字，它们也不会变。Ubuntu 把初始的设备节点名放在紧挨其上的注释行里。下一字段是挂载点（mount point），它很简单，就是文件系统里某个目录。挂载点可以在单独挂载过的某个分区上。

■提示：谨记/etc/fstab 文件里的条目是按从上到下的顺序处理的。

文件系统类型字段（type）告诉系统它的期望类型。如果此项不符，则挂载失败。你可以指定一个用逗号隔开的类型尝试列表，DVD-ROM 驱动器/dev/scd0 就是这样的情况。首先尝试 udf DVD 文件类型，然后是 CD-ROM 所使用的 iso9660。

挂载选项也是以逗号间隔的列表进行传递。在 fstab 的例子中，可以看到对 ext3 文件系统使用了两个不同的选项。relatime 选项用来增强文件系统的性能，这将在第 17 章详细介

绍。另外一个选项 `errors=remount-ro` 控制文件系统出错时的反应。本例中，文件系统在出错时将立即以只读模式挂载。这样就阻止了更多的数据损坏，从而保持文件对服务和用户的可用性。

对于出错时的行为，另外两种可能的值是 `continue` 和 `panic`。前者将使系统写一个日志输入项但会忽略问题，后者将导致系统彻底地崩溃。出错时的默认行为也可通过 `tune2fs -e` 命令在文件系统自身中指定。

有的文件系统不支持 UNIX 风格的文件权限（如 FAT32 和 NTFS），或者可能改变性能。对这些文件系统上文件和目录的访问由其他挂载选项定义。每个所支持的文件系统的选项都可以在 `mount` 手册页里找到。

转存（`dump`）数据字段包含一个数字（0 或 1），它告诉系统是否在执行文件系统检查时转存某些文件系统的元信息。这些转存信息对文件系统修复工具有用。稍后的“故障恢复”一节将更深入介绍它。

最后，`pass` 字段用来确定文件系统的检查顺序。在 `fstab` 文件中，根文件系统被列为 1，因此首先检查它。之后检查 `/boot` 文件系统。此栏为 0 的文件系统最后检查。关于这些字段更长的描述可以到 `fstab` 手册页上查找。

为了添加新分区，你需要知道它的 UUID 号。这可以在 `tune2fs` 的列表中找到，或者使用 `blkid` 实用程序。如果不带任何参数地运行后者，它将打印所有检测到的块设备的 UUID，如列表 8-10 所示。

列表 8-10 显示所有的 UUID

```
$ sudo blkid
/dev/mapper/au--mel--ubuntu--1-root: UUID=" e9e48791-dd89-4bf7-921e-f6d460f7ca24"
    TYPE="ext3"
/dev/mapper/au--mel--ubuntu--1-swap_1: TYPE="swap"
    UUID="83f01bff-2cc4-4f4a-abe1-0b9210c5dfa8"
/dev/sda1: UUID="74bce18f-6f8f-4f4c-be91-4ce38e2bff8e" TYPE="ext3"
/dev/sda5: UUID="pahIPT-TRbM-NB5q-eRWm-G00g-uOli-bf5mOi" TYPE="lvm2pv"
/dev/sdb1: UUID="f06da31f-ac2e-4e66-9f71-81373a36086e" TYPE="ext3" SEC_TYPE="ext2"
/dev/sdb5: UUID="7dd09ce2-cf34-4696-9cdd-eb5420c4f37e" TYPE="jfs"
/dev/sdb6: TYPE="swap" UUID="96570203-0efd-4c18-8887-217d873a6051"
```

为了让它打印单个设备的 UUID，传递设备节点名作为一个参数。

为了以默认的挂载选项挂载我们的 `ext3` 分区，往 `/etc/fstab` 文件里添加下面一行。

```
UUID=f06da31f-ac2e-4e66-9f71-81373a36086e /mnt/data ext3 defaults 0 0
```

注意，必须去除 `blkid` 打印的 UUID 上的引号。如果想用设备节点代替，那么你就可以加下面一句。

```
/dev/sdb1 /mnt/data ext3 defaults 0 0
```

现在你无需重启就可以测试这个输入项。如果使用 `mount` 命令而且只传递挂载点作为参数，它将检查 `/etc/fstab` 文件，找出匹配的条目，进而按照文件中指定的选项挂载它。

```
$ sudo mount /mnt/data
```


如果 mount 命令没打印任何错误就退出，那么这个 fstab 输入项是正确的，我们的文件系统将在每次启动主机时自动挂载。为了复查文件系统的挂载情况，可以运行无任何参数的 mount 命令，它将列出主机上所有挂载的文件系统，如列表 8-11 所示。

列表 8-11 挂载的所有文件系统

```
$ mount
/dev/mapper/au--mel--ubuntu--1-root on / type ext3 (rw,relatime,errors=remount-ro)
proc on /proc type proc (rw,noexec,nosuid,nodev)
/sys on /sys type sysfs (rw,noexec,nosuid,nodev)
varrun on /var/run type tmpfs (rw,noexec,nosuid,nodev,mode=0755)
varlock on /var/lock type tmpfs (rw,noexec,nosuid,nodev,mode=1777)
udev on /dev type tmpfs (rw,mode=0755)
devshm on /dev/shm type tmpfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
/dev/sda1 on /boot type ext3 (rw,relatime)
securityfs on /sys/kernel/security type securityfs (rw)
/dev/sdb1 on /mnt/data type ext3 (rw)
```

另外，为了添加 XFS 或 JFS 分区，我们创建挂载点目录，并往/etc/fstab 添加正确的 UUID（或者设备节点名）和文件系统类型。

```
UUID=7dd09ce2-cf34-4696-9cdd-eb5420c4f37e /mnt/other jfs defaults 0 0
```

注：fstab 文件中的一个错误就可能 导致系统不能启动。如果是这样的话，你可能需要按照“故障恢复”一节中所述的一些步骤进行操作，或者使用第 5 章所描述的单用户模式启动并修复错误。

对内核来说，有可能没有记录到某个文件系统上 UUID 的改变。如果发生了这样的事，尝试用 UUID 的索引挂载文件系统可能会导致下面所示的错误。

```
$ sudo mount /mnt/other
mount: special device /dev/disk/by-uuid/7dd09ce2-cf34-4696-9cdd-eb5420c4f37e
does not exist
```

你可以通过重启 udev 服务使系统重新检测 UUID，并在/dev/disk/by-uuid 中创建正确的符号链接。

提示：为避免键入如 UUID 这样的长字符串，你可以安装 gpm 软件包并运行 gpm 实用程序，这样就可以在终端里使用鼠标复制和粘贴。选定文本复制它，然后粘贴。

8.2.2 检查文件系统的利用率

当开始使用文件系统存储数据时，你将希望能密切注视可用空间大小。当文件系统装满时，使用它的服务可能拒绝启动、停止工作或者崩溃。

通过 df 命令（通常要带-h 选项）可以列出整个文件系统的空间利用率。-h 选项产生人可读的输出，输出以 KB、MB、GB 或 TB 而不是以块为单位，如列表 8-12 所示。

列表 8-12 文件系统利用率

```
$ df -h
Filesystem      Size      Used Avail Use% Mounted on
/dev/mapper/au--mel--ubuntu--1-root
                3.6G      618M    2.8G   19% /
varrun          252M       64K    252M    1% /var/run
varlock         252M        0    252M    0% /var/lock
udev            252M      80K    252M    1% /dev
devshm          252M        0    252M    0% /dev/shm
/dev/sda1       236M      25M    199M   11% /boot
/dev/sdb1       1.9G      35M    1.9G    2% /mnt/data
/dev/sdb5       3.8G     608K    3.8G    1% /mnt/other
```

输出显示了每个已挂载文件系统的总容量、已使用空间容量、可使用空间容量和使用率。此命令执行快速，因为它只是简单地查询文件系统的元数据。

注：要检查索引节点的数量，你可使用 `df i` 命令。使用此命令的情况是：即使明显还有大量的空闲空间，应用程序还是报告磁盘满了。

使用 `du` 命令可以检查一个目录及其所包含的文件的累积容量。此命令需要递归地扫描该目录中的文件，因此需要花费很长一段时间才能完成。我们将再次使用 `-h` 选项以得到人可读的输出。默认情况下，它还将打印每个子目录的大小。为避免这种情况，传递 `-s` 选项，这样它就只显示最终的总数。

```
$ du -sh /etc
du: cannot read directory '/etc/lvm/backup': Permission denied
du: cannot read directory '/etc/lvm/cache': Permission denied
du: cannot read directory '/etc/ssl/private': Permission denied
3.7M    /etc
```

由于扫描目录的缘故，此命令可能遇到我们没有权限访问的目录。这些目录的大小不能计算，因此这种情况下报告的总数不正确。为了总是得到正确的总数，可以 `root` 用户身份运行 `du` 命令。

```
$ sudo du -sh /etc
3.8M    /etc
```

当某个文件系统变满而需要确定把哪些目录移到它们自己的一个分区时，这可能有用。一个备选方案是调整文件系统的大小，我们很快就会谈到这一点。

提示：在第 17 和第 18 章介绍监控和日志时，我们将涉及对文件系统监控的自动化方法。

8.3 RAID

在服务器上用硬盘存储数据以保持它可存取，这很好。但是如果磁盘坏了，数据会丢失。要解决这个问题，你可以使用 RAID。

RAID 允许把多个磁盘当作单个较大磁盘那样使用，有可选的内置冗余度。RAID 的实现有 3 种明显的类型。

- 硬 RAID。
- 仿 RAID。
- 软 RAID。

硬 RAID 使用专门的硬件控制器，通常被称为“RAID 控制器”，它管理 RAID，对操作系统透明。企业级服务器常常带有这些专用硬件控制器。在这类系统上，通常通过 BIOS 配置 RAID（在第 5 章简要地介绍过）。那么 Linux 看到的将是单个 RAID 阵列，使用起来就像一个普通的硬盘。

仿 RAID 是硬 RAID 的小型版，用于较小的系统或者桌面机。现在制造商可能已经通过一个芯片在主板上增加了 RAID 功能。我们不推荐使用仿 RAID，因为这种实现方式下的 RAID 阵列只能在共享同一个控制器的主机上工作。它的性能还依赖于制造商所提供的有知识产权的代码。这些控制器一般被配置为类似串行 ATA 控制器而不是 RAID 的运行方式。最常见的仿 RAID 控制器的短列表见 <http://linux-ata.org/faq-sata-raid.html>。

■注：如果用 Windows 配置了仿 RAID 并且想双启动，通过 dmraid 系统大多数仿 RAID 阵列在 Linux 下仍然可以使用。停掉仿 RAID 可能引起 Windows 不工作，数据也可能丢失。

第三种 RAID 实现类型是通过包含在 Linux 内核中的软件实现的。这种系统被称为 md 系统或多磁盘系统。md 系统通常比仿 RAID 性能更好，并且 md 系统的 RAID 阵列可在主机之间传递。本节将关注 md 系统的 RAID 的使用。

8.3.1 RAID 的类型

有好几种类型（或者等级）的 RAID。使用哪个等级取决于什么对你最重要。这些不同的等级在磁盘空间、可靠性和速度等方面提供了权衡的余地。表 8-4 列出了最常用的 RAID 等级。

表 8-4 常用的 RAID 等级		
RAID 等级	功能	存储容量
RAID 0	速度	$N \times \text{size}$
RAID 1	冗余度	$N \times \text{size} / 2$
RAID 5	冗余度、速度	$(N - 1) \times \text{size}$
RAID 6	冗余度、可靠性、速度	$(N - 1) \times \text{size}$
RAID 10	冗余度、可靠性、速度	$(N / 2) \times \text{size}$

要查找 RAID 等级的详尽列表和描述，可访问 http://en.wikipedia.org/wiki/Redundant_array_of_independent_disks。

使用一个硬盘作为备用也很常见。如果阵列中某个磁盘不能工作了，它的位置可立刻由备用磁盘接管。

■注：运行没有任何备用设备的 RAID 是可能的，但为了避免数据丢失，需要到时立刻替换坏掉的设备。

1. 分拆式和镜像式磁盘

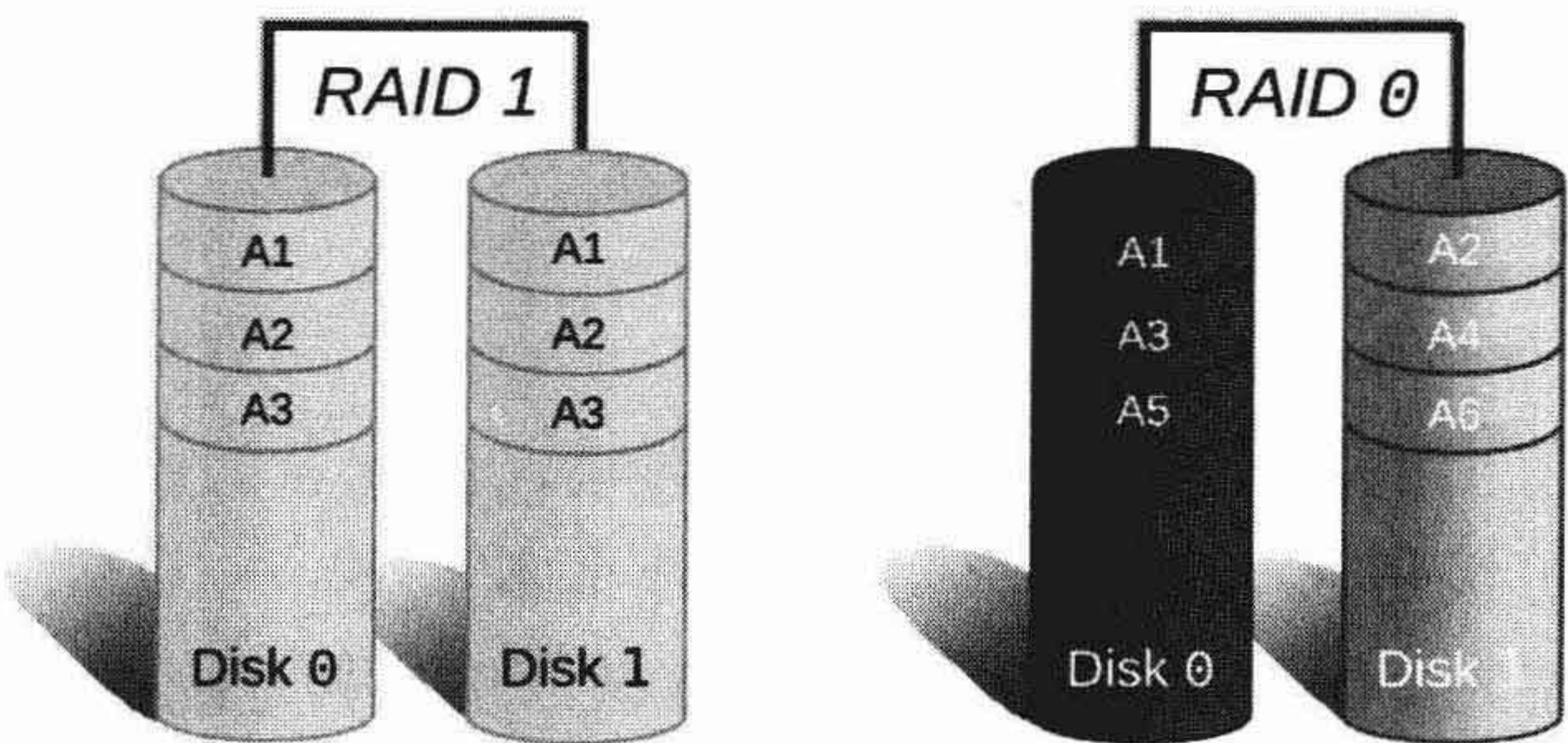
RAID 最基本的使用方式是用两个磁盘，可选择使用 RAID 等级 0 或 RAID 等级 1。

RAID 0 也称“分拆式磁盘” (stripping)，它的两个磁盘在 Linux 下看就是一个两倍大的组合式磁盘。当向这种 RAID 阵列写数据时，数据的各个部分将散落在每一个磁盘里。因为 Linux 可同时在两个磁盘上执行这种操作，所以写入 RAID 0 就比写入单个磁盘快。不过缺点是：当其中一个磁盘坏了，遍布在两个磁盘的任意部分的文件都将消失，这样所有的数据都丢失了。

■注：在服务器或者任何保存非短期数据的机器上，要避免使用 RAID 0。

RAID 1 也称“镜像式磁盘” (mirroring)，它允许在阵列里存储的数据量只能是单个磁盘能保存的那么多。它在两个磁盘上都同样存储了所有文件的拷贝。因此，如果一个磁盘坏了，仍然可从另一个中找回。因为所有数据都需要写入每个磁盘，所以 RAID 1 不能提供较好的写性能。

图 8-1 显示了如何使用 RAID 0 或 RAID 1 存储文件。在 RAID 1 上，每个磁盘都包含每个文件的完整拷贝。在 RAID 0 上，每个磁盘只包含每个文件的一部分拷贝。



Disk 0: 磁盘 0; Disk 1: 磁盘 1

图 8-1 RAID 0 和 RAID 1 的文件存储

当有更多磁盘可供利用时，我们将有更多选项来挑选一个既有冗余度又有较好性能的 RAID 等级。最简单的扩展是 RAID 1+0 (RAID 10)，它使用多个 RAID 1 镜像磁盘作为 RAID 0 分拆式磁盘的基本元素。于是所有分拆的数据至少保存在两个磁盘上，优点是既有 RAID 0 的速度又有 RAID 1 的冗余度。不过，能存储的数据量只有总的磁盘容量的一半。如果磁盘既大又便宜，并且服务器上有足够的插槽支持至少 4 个磁盘，那么这也许是一个不错的选择。

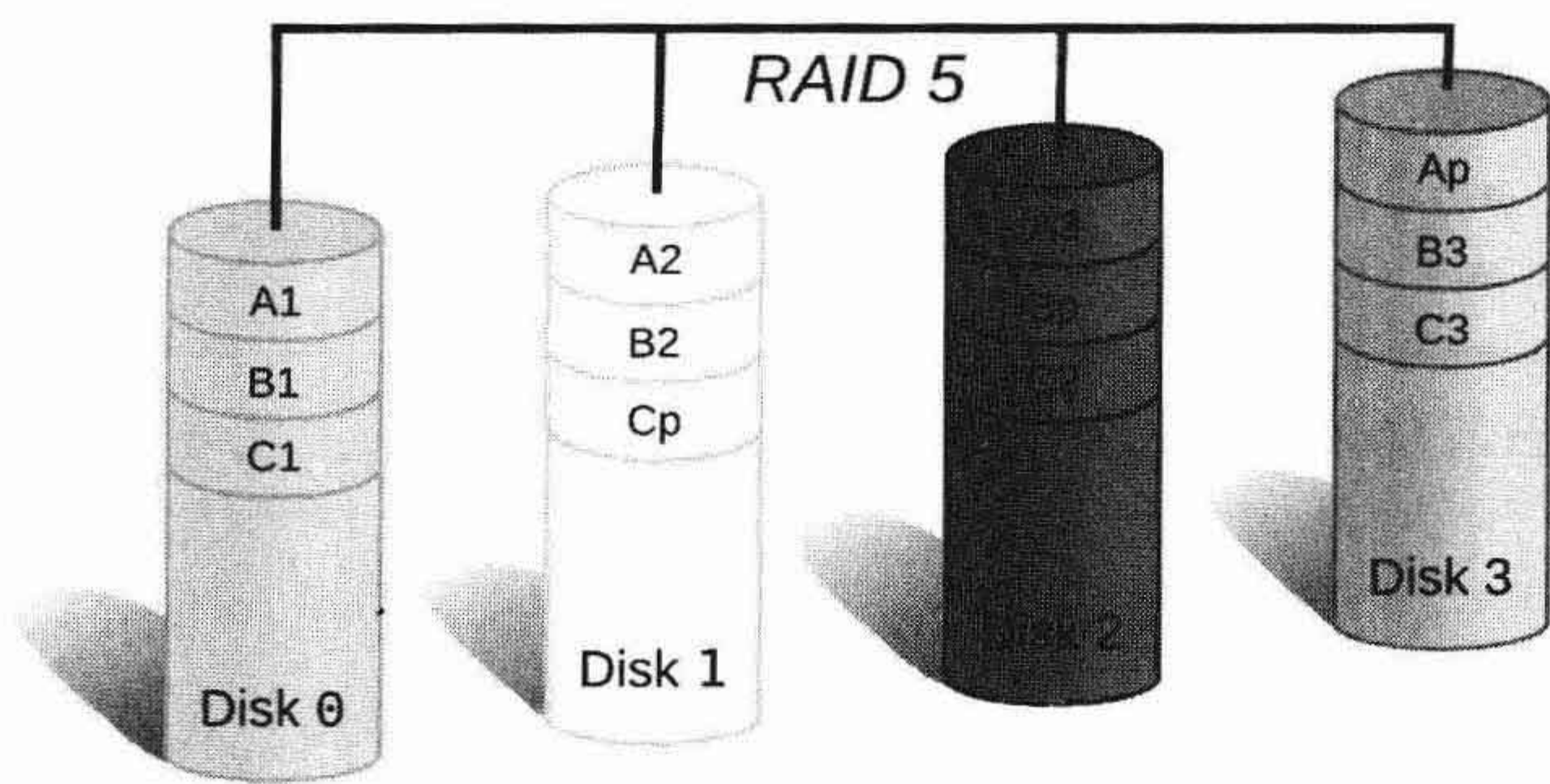
2. 处理器来帮忙

为了在冗余度、存储量和速度等方面尽善尽美，你可以寻求某些处理能力的帮助。RAID 5 至少使用 3 个磁盘。它使存储空间的使用效率更高，读写速度更快。这是因为它把数据分

拆到多个磁盘，还把每个条纹的校验和写入另一个磁盘。如果有一个磁盘坏了，校验和就可用来重建丢失条纹里的数据。

这种方法的代价是需要有计算校验和的处理能力。当数据被写入阵列时，必须计算校验和并存储在其中一个磁盘里。如果一个磁盘坏了，就可以接着用检验和与没损坏的磁盘上的数据重新计算丢失的那部分数据。CPU 越快，上述处理就越快。

图 8-2 给出一个简单的示意图，描述了数据与校验和如何分拆到磁盘上。B1、B2 和 B3 是文件 B 的组成部分。Bp 是校验和。如果磁盘 1 坏了，B2 就可以从 B1、B3 和 Bp 计算得来。于是当替代磁盘增加到位时，它的内容就可以恢复了。



Disk 0: 磁盘 0; Disk 1: 磁盘 1; Disk 2: 磁盘 2; Disk 3: 磁盘 3

图 8-2 RAID 5 在多个磁盘上的条纹布局

重要的是要牢记，使用 RAID 不能替代创建常规备份。它可以在硬件故障中保护数据，但不能在人为删除后恢复数据。如果 RAID 阵列中某个文件被意外删除，那么阵列中所有设备都将移除它。第 12 章将介绍数据备份与恢复。

8.3.2 创建阵列

我们想要保护主机上的数据免受磁盘故障的影响，因此希望在磁盘上使用 RAID。服务器上最常用的 RAID 等级是 RAID 1 和 RAID 5，那么本节将一一设置它们。用哪一个取决于有多少可用的磁盘。首先，你要确保至少有 3 个磁盘，并在所有磁盘上创建同等尺寸的分区。

■注：如果没有足够的磁盘作 RAID，你可在单个磁盘上创建多个同样大小的分区，使用它们作为 RAID 阵列的组件。这样可以测试 RAID 的安装和管理。注意这种配置下的性能相当低，因为数据需要被多次往同一磁盘的各个不同部分写入。它也不能提供更多额外的弹性来应对磁盘故障。如果磁盘坏了，那么整个 RAID 阵列也就坏了。

在我们的实例主机中，将使用 sdb（因而应该从/etc/fstab 文件中移除此磁盘上的文件系统）并增加两个相同的附加磁盘：sdc 和 sdd。分区类型应该设置为 da-Non-FS data，如列表 8-13 所示。

列表 8-13 清除分区表并创建一个 RAID 分区

```
$ sudo fdisk /dev/sdb
```

```
The number of cylinders for this disk is set to 1044.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
```


- 1) software that runs at boot time (e.g., old versions of LILO)
- 2) booting and partitioning software from other OSs
(e.g., DOS FDISK, OS/2 FDISK)

Command (m for help): o
Building a new DOS disklabel with disk identifier 0x2bcf6dd1.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

The number of cylinders for this disk is set to 1044.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:

- 1) software that runs at boot time (e.g., old versions of LILO)
- 2) booting and partitioning software from other OSs
(e.g., DOS FDISK, OS/2 FDISK)

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

Command (m for help): n

Command action

e extended

p primary partition (1-4)

p

Partition number (1-4):

Value out of range.

Partition number (1-4): 1

First cylinder (1-1044, default 1):

Using default value 1

Last cylinder or +size or +sizeM or +sizeK (1-1044, default 1044):

Using default value 1044

Command (m for help): t

Selected partition 1

Hex code (type L to list codes): da

Changed system type of partition 1 to da (Non-FS data)

Command (m for help): w

The partition table has been altered!

Calling ioctl() to re-read partition table.

Syncing disks.

对/dev/sdc 和 dev/sdd 重复上述过程。对它们可以省掉第一步：o 命令。此命令擦除当前的分区表。

既然 3 个磁盘都已准备好，你就可以创建 RAID 阵列了。为此，你要用到 RAID 管理程序，此程序由 mdadm 软件包提供。

管理 RAID 配置的命令也叫 mdadm，通过模式选项可以指明它应该对阵列进行的操作。为了创建阵列，你需要指定 create 模式、想要使用的 RAID 等级和需要成为阵列组成部分的那些分区。列表 8-14 显示了创建 RAID 1 阵列的方式。

列表 8-14 创建带热备份的 RAID 1 阵列

```
$ sudo mdadm --create /dev/md0 --level=raid1 --raid-devices=2 /dev/sdb1 ➡
/dev/sdc1 --spare-devices=1 /dev/sdd1
mdadm: /dev/sdb1 appears to contain an ext2fs file system
size=1959896K mtime=Mon Dec 29 19:04:24 2008
```



```
Continue creating array? y
mdadm: array /dev/md0 started.
```


因为 fdisk 只影响分区表，所以早前创建的文件系统根本不受影响，它们的数据仍然存在磁盘里。既然你不再使用它们，那么就告诉 mdadm 继续创建阵列。

创建或启动 RAID 阵列将引起 md 核心模块的载入并显示一些状态信息。通过 dmesg 你可以检查内核日志，如列表 8-15 所示。

列表 8-15 内核日志中的 RAID 信息

```
$ dmesg
[58663.877141] md: bind<sdb1>
[58663.902844] md: bind<sdcl>
[58663.942803] md: bind<sdd1>
[58663.943742] md: md0: raid array is not clean -- starting background
reconstruction
[58663.992313] md: raid1 personality registered for level 1
[58663.994631] raid1: raid set md0 active with 2 out of 2 mirrors
[58664.027163] md: resync of RAID array md0
[58664.027196] md: minimum _guaranteed_ speed: 1000 KB/sec/disk.
[58664.027224] md: using maximum available idle IO bandwidth (but not more than
200000 KB/sec) for resync.
[58664.027543] md: using 128k window, over a total of 8385792 blocks.
[58705.830365] md: md0: resync done.
[58705.863620] RAID1 conf printout:
[58705.863646] --- wd:2 rd:2
[58705.863700] disk 0, wo:0, o:1, dev:sdb1
[58705.863716] disk 1, wo:0, o:1, dev:sdcl
```

因为新阵列还没有同步过，所以内核将首先确保两个磁盘上的数据相同。它表示同步过程将尽可能快地进行，每个磁盘的速度绝不低于 1000KB/s，总速度不高于 200000KB/s。

 **提示：**第 17 章将说明如何改变同步速度。

为了检查该 RAID 设备的状态，你可以使用查询模式下的 mdadm 程序，并使用--detail 选项。它显示很多有关指定 RAID 设备的信息，如列表 8-16 所示。

列表 8-16 查询 RAID 设备状态

```
$ sudo mdadm --query --detail /dev/md0
/dev/md0:
    Version : 00.90.03
  Creation Time : Tue Dec 30 11:13:43 2008
    Raid Level : raid1
    Array Size : 8385792 (8.00 GiB 8.59 GB)
  Used Dev Size : 8385792 (8.00 GiB 8.59 GB)
    Raid Devices : 2
   Total Devices : 3
Preferred Minor : 0
  Persistence : Superblock is persistent

               Update Time : Tue Dec 30 11:21:03 2008
                 State : clean
```



```
Active Devices : 2
Working Devices : 3
Failed Devices : 0
Spare Devices : 1

        UUID : 06b4cbbc:3838c8c8:9cc6715a:1cc4c6dd (local to host au-mel-ubuntu-1)
        Events : 0.4

Number      Major      Minor      RaidDevice State
    0         8         17          0    active sync /dev/sdb1
    1         8         33          1    active sync /dev/sdc1
    2         8         49         -    spare   /dev/sdd1
```

列表 8-16 显示了阵列相关的元信息，还有每个组件的详细状态。以 RAID 1 阵列为例，你可以看到/dev/sdb1 和/dev/sdc1 都是在活动和同步的状态。这意味着任何写入 RAID 设备的数据都立刻写入/dev/sdb1 和/dev/sdc1 中。如果这些设备中的任何一个坏了，备用盘 (/dev/sdd1) 将自动激活并同步。

在开机时，Linux 主机调用 mdadm 程序。此程序将根据它的配置，扫描全部分区或者所规定的磁盘以寻找 RAID 超级块。如果发现情况，它将分析它们并试图装配和启动所有 RAID 阵列。你还可以在 mdadm 配置文件中明确定义 RAID 阵列，以确保它们的设备节点名不改变。定义阵列的配置文件在 Red Hat 里是/etc/mdadm.conf，在 Ubuntu 里是/etc/mdadm/mdadm.conf。列表 8-17 包含了一个来自 Ubuntu 系统的默认 mdadm.conf 文件。

列表 8-17 默认的配置文 件 mdadm.conf

```
# mdadm.conf
#
# Please refer to mdadm.conf(5) for information about this file.
#

# by default, scan all partitions (/proc/partitions) for MD superblocks.
# alternatively, specify devices to scan, using wildcards if desired.
DEVICE partitions

# auto-create devices with Debian standard permissions
CREATE owner=root group=disk mode=0660 auto=yes

# automatically tag new arrays as belonging to the local system
HOMEHOST <system>

# instruct the monitoring daemon where to send mail alerts
MAILADDR root

# definitions of existing MD arrays

# This file was auto-generated on Tue, 30 Dec 2008 11:02:03 +1100
# by mkconf $Id$
```

这个配置文件将导致主机在启动时扫描阵列并创建由 root 用户和 disk 组所拥有的设备节点，就像常规的硬盘驱动器那样。它规定，当 mdadm 在监控模式下运行时，任何有关设备故障的电子邮件都将发送给 root 用户。

■注：第 10 章将说明如何把所有给 root 用户的电子邮件重定向到另一个地址。

最后，有一块空间可以添加 RAID 阵列的这些配置。在这里为 RAID 1 阵列加上一个定义。根据 mdadm.conf 手册页，你需要如下这样做。

```
ARRAY /dev/md0 level=raid1 num-devices=2 spares=1 ➡
    UUID=06b4cbbc:3838c8c8:9cc6715a:1cc4c6dd devices=/dev/sdb1,/dev/sdc1,/dev/sdd1
```

注意，严格意义上，为此阵列添加定义不是必须的。即使在配置文件中没有考虑阵列定义，mdadm 也会自动检测并装配阵列。

我们已提到，如果有事件发生，mdadm 会发一封电子邮件给 root 用户。稍后会给出一个例子，那时将开始舍弃阵列里的设备。为了快速地检查所有 RAID 阵列的状态，你可以 cat 文件 /proc/mdstat 里的内容。

```
$ cat /proc/mdstat
Personalities : [raid1]
md0 : active raid1 sdd1[2](S) sdc1[1] sdb1[0]
      8385792 blocks [2/2] [UU]

unused devices: <none>
```

这会显示所有 RAID 阵列及其全部组件的状态。sdd1 后的 (S) 说明此设备将作为备用设备使用。

/proc 文件系统

你可能已经注意到本章多次提及被称为 /proc 的目录。这是一个包含虚拟文件的特殊目录，这些虚拟文件提供一种与内核交互的方式。例如，通过 cat /proc/cpuinfo 可以得到关于主机处理器的信息。

通过目录 /proc/sys 下的文件，你可以访问内核的内部变量。第 17 章将介绍这些变量。

/proc 文件系统并不真的存在于磁盘里，因此它不占用任何空间，尽管它好像包含一些很大的文件。

要了解更多关于 /proc 文件系统的信息，可以访问 <http://en.wikipedia.org/wiki/Procfs> 和 http://www.redhat.com/docs/manuals/enterprise/RHEL-4-Manual/en-US/Reference_Guide/ch-proc.html。

如果拥有最低要求的 4 个硬盘，那么可改为创建 RAID 5 阵列。这样做可更有效地利用存储空间，而且在某些情况下还可以提高性能。为了创建新的 RAID 5 阵列，你需要首先拆卸 RAID 1 阵列。停掉它就释放了它所用的所有设备。

```
$ sudo mdadm --manage /dev/md0 --stop
mdadm: stopped /dev/md0
```

现在就可以把这些设备用到新的 RAID 5 阵列中了。注意还要移除原来添加到 mdadm.conf 文件中的相关条目。我们将添加一个磁盘 /dev/sde，并在其上创建一个单独的类型为 da 的主分区。

当上述工作完成后，你就可以创建一个有 3 个活动设备和 1 个备用设备的 RAID 5 阵列，如列表 8-18 所示。

列表 8-18 创建 RAID 5 阵列

```
$ sudo mdadm --create /dev/md0 --level=raid5 --raid-devices=3 --spare-devices=1 ➡
/dev/sdb1 /dev/sdc1 /dev/sdd1 /dev/sde1
mdadm: /dev/sdb1 appears to contain an ext2fs file system
size=1959896K mtime=Mon Dec 29 19:04:24 2008
mdadm: /dev/sdb1 appears to be part of a raid array:
level=raid1 devices=4 ctime=Tue Dec 30 11:13:43 2008
mdadm: /dev/sdc1 appears to contain an ext2fs file system
size=1959896K mtime=Mon Dec 29 19:04:24 2008
mdadm: /dev/sdc1 appears to be part of a raid array:
level=raid1 devices=4 ctime=Tue Dec 30 11:13:43 2008
mdadm: /dev/sdd1 appears to contain an ext2fs file system
size=1959896K mtime=Mon Dec 29 19:04:24 2008
mdadm: /dev/sdd1 appears to be part of a raid array:
level=raid1 devices=4 ctime=Tue Dec 30 11:13:43 2008
Continue creating array? y
mdadm: array /dev/md0 started.
```

在这些设备中，有些曾是以前 RAID 1 阵列的一部分，它们仍然包含那些带有阵列信息的旧 RAID 超级块。因为要创建新的阵列覆盖旧数据，所以回答 Y。原来 RAID 1 阵列是同步过的，因此，即使当初只在 `/dev/sdb1` 上创建了一个文件系统，现在所有设备还是包含此文件系统。你可以通过 `/proc/mdstat` 再检查一次阵列状态。

```
$ cat /proc/mdstat
Personalities : [linear] [multipath] [raid0] [raid1] [raid6] [raid5] [raid4]
               [raid10]
md0 : active raid5 sdd1[2] sde1[3](S) sdc1[1] sdb1[0]
      16771584 blocks level 5, 64k chunk, algorithm 2 [3/3] [UUU]

unused devices: <none>
```

现在就有了带 3 个活动设备和 1 个备用设备的 RAID 5 阵列。假如拥有足够的硬盘，你还可以增加 RAID 5 阵列的规模。这样会引起数据的移位以及校验和的重算，因而将花费一些时间来完成。往主机增加第 6 个磁盘 `/dev/sdf`，这样就可以增加阵列的规模，并且仍然有一个可用的备用设备。你还需要对这个磁盘分区，与别的磁盘一样，使它只包含一个单独的 `da` 类型的主分区。接着你就可以使用管理模式下的 `mdadm`，通过 `--add` 选项往阵列里添加新设备，进而增加阵列规模，如列表 8-19 所示。

列表 8-19 扩展 RAID 5 阵列

```
$ sudo mdadm --manage /dev/md0 --add /dev/sdf1
mdadm: added /dev/sdf1
```

为了在一条命令里完成多个设备的添加，你只需在 `--add` 选项后面把它们全部列出。现在快速检查 `/proc/mdstat`，结果显示 `sde1` 和 `sdf1` 两者都被列为备用设备。

```
$ cat /proc/mdstat
Personalities : [linear] [multipath] [raid0] [raid1] [raid6] [raid5] [raid4]
               [raid10]
md0 : active raid5 sdf1[3](S) sdb1[0] sde1[4](S) sdd1[2] sdc1[1]
      16771584 blocks level 5, 64k chunk, algorithm 2 [3/3] [UUU]

unused devices: <none>
```


现在你可以使用增长模式下的 `mdadm` 把阵列的活动磁盘从 3 个扩展到 4 个。备用磁盘中的一个将自动用作活动磁盘。这个过程的部分有破坏性，因此，如果在扩展阵列时发生电源故障，可能会丢失数据。为了预防这种事情，你可指定 `--backup-file` 选项。你要保证不把备份文件保存到 `/tmp` 目录中，因为主机启动时会清空这个目录！

```
$ sudo mdadm --grow /dev/md0 --raid-disks=4 --backup-file=/root/raid-backup-file
mdadm: Need to backup 384K of critical section..
mdadm: ... critical section passed.
```

你可以再次通过 `/proc/mdstat` 文件密切关注此进程。

```
$ cat /proc/mdstat
Personalities : [linear] [multipath] [raid0] [raid1] [raid6] [raid5] [raid4]
               [raid10]
md0 : active raid5 sdf1[3] sdb1[0] sde1[4] (S) sdd1[2] sdc1[1]
      16771584 blocks super 0.91 level 5, 64k chunk, algorithm 2 [4/4] [UUUU]
      [=====>.....] reshape = 35.8% (3008512/8385792) finish=22.2min
      speed=4022K/sec

unused devices: <none>
```

现在有 4 个活动设备和 1 个单独的备用设备。阵列全新的容量只有在改造完成后才可访问。正如所见，改造运行的速率远比 RAID 1 的同步低。

■ **提示：**除了手动重新运行 `cat /proc/mdstat` 命令外，你还可以通过 `watch` 命令指定一个时间间隔使它自动运行。登入另一个控制台并运行 `watch -n 5 cat /proc/mdstat`，它使上述命令每 5 秒就自动运行一次。按 `Ctrl+C` 键可以退出 `watch`。

本章“故障恢复”一节将重回到 RAID 并说明如何处理磁盘故障。要了解更多有关 Linux 上 md RAID 的信息，可以访问 http://linux-raid.osdl.org/index.php/Linux_Raid。

接下来将介绍无需分区就可利用这些 RAID 设备的方法。

8.4 逻辑卷管理

用分区划分磁盘是很好，但是它们太不灵活。一旦对磁盘分了区，就很难调整分区的大小或者追加分区。即使增加了磁盘和分区，数据可能变得遍布各个位置和目录。这样就更难合并、备份和管理数据，还可能使用户更难查找他们的数据。为了克服这个问题，人们创造了逻辑卷管理（Logical Volume Management, LVM）的概念。

LVM 不是把磁盘分成固定数量的分区，并把它们存放在磁盘的固定区域，而是把一个或多个分区合成一个单独的逻辑卷组。在卷组里可以动态地创建、调整和删除卷，这样就没有卸载卷或重启系统以更新分区图的必要了。

LVM 系统有三层。底层由物理卷（磁盘、分区或 RAID 阵列）组成。物理卷用来创建卷组。每个卷组可包含一个或多个物理卷。最后，卷组可以包含任意数目的逻辑卷，而逻辑卷是 LVM 中与分区对等的概念。

Ubuntu 和 Red Hat 两者都有可用于管理 LVM 的图形化工具。不过，首先介绍使用命令行的管理，就从 LVM 卷和卷组开始。

8.4.1 创建卷与卷组

如果想以 LVM 的方式使用某个分区，你需要用 `fdisk` 把它的类型设置为 `8e-Linux LVM`。你也可以使用整个磁盘或 RAID 阵列作为 LVM 的存储。这很方便，因为通常不会在软 RAID 阵列上创建分区。要在 LVM 中使用整个磁盘，需要先擦除它的分区表。这可通过使用 `dd` 命令往磁盘的第一扇区（前 512 字节）写 0 来完成，如下所示。

```
$ sudo dd if=/dev/zero of=/dev/sdg bs=512 count=1
```

此命令从 `/dev/zero` 读取单块的 512 字节数据，然后把它写到 `/dev/sdg` 的第一扇区，从而擦除了启动记录和分区图，我们知道它们就存储在那儿。在 `dd` 手册页可以了解更多关于 `dd` 实用程序的信息。

注：`dd` 命令是非常强大的实用程序，可以写裸块设备，并且可以轻易地毁掉数据。使用它要极其小心。我们有时也用它创建分区或其他块设备的备份。

在我们的例子中，将对早前创建的 RAID 5 阵列配置 LVM。配置步骤在 RAID 1、单独的分区或者整个磁盘上都是一样的。

LVM 使用的这些存储设备每一个都被称为一个物理卷（Physical Volume, PV）。通过 `pvcreate` 命令你可以把某个设备标记为物理卷。

```
$ sudo pvcreate /dev/md0
Physical volume "/dev/md0" successfully created
```

此命令在设备的开头打了一个小水印，以使 LVM 能识别它。通过 `pvs` 命令你可列出系统里所有这样的设备。

```
$ sudo pvs
  PV          VG          Fmt  Attr  PSize  PFree
/dev/md0      lvm2  --    23.99G 23.99G
/dev/sda5    au-mel-ubuntu-1  lvm2  a-    3.76G   0
```

注：如果使用 `pvdisplay` 命令，你可以得到有关物理卷的更详细信息。

记得当初安装系统时，我们选择使用 LVM。现在你可以看到它把 `/dev/sda5` 分区用作物理卷。第二栏标注为 `VG`，指的是卷组，它在 LVM 系统的下一层。

通过 `vgs` 命令你可以列出系统上所有的卷组。

```
$ sudo vgs
VG          #PV #LV #SN Attr   VSize  VFree
au-mel-ubuntu-1  1  2  0  wz--n- 3.76G   0
```

注：如果使用 `vgdisplay` 命令，你可以得到有关卷组的更详细信息。

有一个名为 `au-mel-ubuntu-1` 的卷组，它是由安装程序创建的。它大小为 1 个物理卷，包括 2 个逻辑卷。通过 `lvs` 命令你可以列出这些卷。目前有两个卷：`root` 和 `swap_1`。

```
$ sudo lvs
LV      VG          Attr      LSize   Origin  Snap%   Move   Log   Copy%
root    au-mel-ubuntu-1 -wi-ao    3.54G
swap_1  au-mel-ubuntu-1 -wi-ao    224.00M
```

注：如果使用 `lvdisplay` 命令，你可以得到有关逻辑卷的更详细信息。

你现在可通过 `vgextend` 命令往已存在的组里添加物理卷。

```
$ sudo vgextend ubuntu-au-mel-1 /dev/md0
Volume group "au-mel-ubuntu-1" successfully extended
```

要检查一下，你可使用 `pvs` 和 `vgs` 显示这些物理卷。

```
$ sudo pvs
PV      VG          Fmt  Attr  PSize  PFree
/dev/md0 au-mel-ubuntu-1 lvm2  a-    23.99G 23.99G
/dev/sda5 au-mel-ubuntu-1 lvm2  a-     3.76G  0

$ sudo vgs
VG          #PV  #LV  #SN  Attr   VSize  VFree
au-mel-ubuntu-1 2    2    0   wz--n- 27.75G 23.99G
```

新的物理卷现在已是 `au-mel-ubuntu-1` 卷组的一部分。向该卷组添加这个新物理卷意味着它现在有 23.99GiB 的未分配空间。

另一种选择是创建一个新卷组并把我们的物理卷包含于其中。为此，你仍可通过 `vgreduce` 命令把 `/dev/md0` 从 `au-mel-ubuntu-1` 移除。

```
$ sudo vgreduce au-mel-ubuntu-1 /dev/md0
Removed "/dev/md0" from volume group "au-mel-ubuntu-1"
```

它现在就可用于另一个卷组了。新卷组可使用 `vgcreate` 命令创建。我们要把 `/dev/md0` 设备分配给新卷组 `raid-volume`。这些完成后，你可使用 `vgs` 命令查看新卷组。

```
$ sudo vgcreate raid-volume /dev/md0
Volume group "raid-volume" successfully created

$ sudo vgs
VG          #PV  #LV  #SN  Attr   VSize  VFree
au-mel-ubuntu-1 1    2    0   wz--n-  3.76G    0
raid-volume    1    0    0   wz--n- 23.99G 23.99G
```

现在就有了一个新卷组，可用来创建逻辑卷。为了给随后的章节做准备，为什么不创建一块用于网站的存储区呢？有了 LVM，你可以轻松地对这些功能分配专用的磁盘区域。

首先，你需要通过 `lvcreate` 命令创建逻辑卷。你要指定名字、大小和创建此卷所在的卷组。然后你可以通过 `lvs` 列出它。

```
$ sudo lvcreate --name www --size 2G raid-volume
Logical volume "www" created

$ sudo lvs
LV      VG          Attr      LSize   Origin  Snap%   Move   Log   Copy%
```



```

root      au-mel-ubuntu-1  -wi-ao      3.54G
swap_1    au-mel-ubuntu-1  -wi-ao      224.00M
www       raid-volume      -wi-a-       2.00G

```

现在剩下来要做的就是在该逻辑卷上创建文件系统并把它挂载到某处。为此，我们需要该逻辑卷的设备节点名。这由一个被称为 **device-mapper** 的驱动程序管理，我们所创建的所有卷的设备节点输入项都由这个程序创建。

逻辑卷可通过 `/dev/mapper/<vgname>-<lvname>` 访问，它是 `/dev/<vgname>/<lvname>` 的符号链接。因此，对于新的“WWW”逻辑卷，你可以使用 `/dev/raid-volume/www`。因为网站很可能由很多较小的文件组成，所以我们使用下面的命令为它创建 **ext3** 文件系统。

```
$ sudo mkfs.ext3 /dev/raid-volume/www
```

你可以把这些卷看作是普通分区一样使用，并把它加到 `/etc/fstab` 文件里，这样主机启动时就会把它自动挂载到 `/srv/www`。为此，你可以使用刚创建的文件系统的设备节点名或 **UUID**。这两者都可由 **blkid** 提供。`/etc/fstab` 里的输入项看起来如下所示。

```

# /dev/mapper/raid--volume-www
UUID=c0f3d1ef-c201-4fae-bb35-6a21bfcafa0e /srv/www ext3 defaults 0 0

```

创建完 `/srv/www` 目录，重启 **udev**，这样新的 **UUID** 就被检测并登记在 `/dev` 下。然后你就可用下面的命令挂载它。

```
$ sudo mount /srv/www
```

8.4.2 扩充逻辑卷

迄今为止，使用 **LVM** 似乎不过是一种更复杂的磁盘使用方式，但是如果想把网站增加到 **2GiB** 呢？没有 **LVM** 的话，你可能需要在未使用的磁盘上创建一个分区，然后把所有数据复制过去，同时确保 `/etc/fstab` 已经更新过。不过，使用 **LVM** 就可以简单地扩充逻辑卷，然后调整其上的文件系统大小。如果卷组里没有剩余空间，你可以先往卷组增加一个物理卷。

为了安全地调整逻辑卷中文件系统的大小，你需要完成两步。首先，你要调整逻辑卷自身的大小；其次，你要调整文件系统的大小。

扩充卷可以使用 **lvextend** 命令。你需要指定新的总量或者增加的量，以及卷名。通过对容量参数增加前缀 **+**，就表明你想往已有容量增加指定的容量。

```
$ sudo lvextend --size +2G /dev/raid-volume/www
```

要指定新的总量，你可以使用下面的命令。

```

$ sudo lvextend --size 4G /dev/raid-volume/www
Extending logical volume www to 4.00 GB
Logical volume www successfully resized

```

在本例中，两种方法产生相同的结果：逻辑卷大小为 **4GB**，并且包含一个 **2GB** 的文件系统。

现在要告诉文件系统它所在设备的新容量，为此，我们使用 **resize2fs** 实用程序。我们要从 **2GB** 的文件系统开始，如列表 8-20 所示。

列表 8-20 调整 ext3 文件系统的大小

```
$ df -h /srv/www
Filesystem                Size  Used Avail Use% Mounted on
/dev/mapper/raid--volume-www
                        2.0G  135M  1.9G   7%   /srv/www

$ sudo resize2fs /dev/raid-volume/www
resize2fs 1.40.8 (13-Mar-2008)
Filesystem at /dev/raid-volume/www is mounted on /srv/www; on-line resizing required
old desc_blocks = 1, new_desc_blocks = 1
Performing an on-line resize of /dev/raid-volume/www to 1048576 (4k) blocks.
The filesystem on /dev/raid-volume/www is now 1048576 blocks long.

$ df -h /srv/www
Filesystem                Size  Used Avail Use% Mounted on
/dev/mapper/raid--volume-www
                        4.0G  137M  3.8G   4%   /srv/www
```

现在就有了 4GB 的文件系统。

XFS 文件系统也可以在使用中扩充。你也需要先扩充逻辑卷，然后再扩充文件系统。对 XFS 要使用 `xfs_grow /srv/www`。此命令用挂载点作为参数，而不是设备节点名。你可以查看 `xfs_grow` 的手册页面以获得更多信息。

8.4.3 缩减逻辑卷

除了扩充，你还可以缩减文件系统。为了缩减文件和逻辑卷，你要使用 `lvreduce` 命令，并按照上述步骤倒序执行。你要确定不要把逻辑卷缩减得比它包含的文件系统还小。

■注：与 ext2、ext3 和 ext4 不同，XFS 文件系统不能缩减。

尽管比简单的分区多了一点配置工作，但是 LVM 使存储空间的使用方式灵活很多。表 8-5 列出了最常用的 LVM 命令。

表 8-5	LVM 基本命令
命令	用途
<code>pvcreat</code>	标记用于 LVM 的设备
<code>pvremove</code>	移除物理卷上的 LVM 标记
<code>pvdisplay/pvs</code>	显示系统上指定设备或所有物理卷的信息
<code>vgcreate</code>	创建新卷组
<code>vgremove</code>	移除（删除）卷组
<code>vgextend</code>	添加物理卷到卷组
<code>vgreduce</code>	移除卷组中的物理卷
<code>vgdisplay/vgs</code>	显示系统上指定组或所有卷组的信息
<code>lvcreate</code>	创建新逻辑卷
<code>lvremove</code>	移除（删除）逻辑卷
<code>lvextend</code>	增加逻辑卷的容量
<code>lvreduce</code>	减少逻辑卷的容量
<code>lvdisplay/lvs</code>	显示系统或指定卷组上的所有逻辑卷

8.4.4 用 GUI 管理 LVM

正如前面简要提到的那样，基于 Red Hat 的主机可通过 GUI 的方式，通过 system-config-lvm 工具管理 LVM。为了能使用此工具，你需要已安装 system-config-lvm 软件包。

注：Ubuntu 8.04 LTS 没有 system-config-lvm 图形化工具。但是在较新版本 Ubuntu 的软件资料库中有这个工具。如果使用 Ubuntu 上的 KDE，你也可以使用执行类似功能的 kvgm 工具。

开启 system-config-lvm 应用程序的步骤是：选择“系统”→“管理”→“逻辑卷管理”或者通过命令行输入下面的命令。

```
$ sudo system-config-lvm
```

图 8-3 显示了这个 GUI 工具。

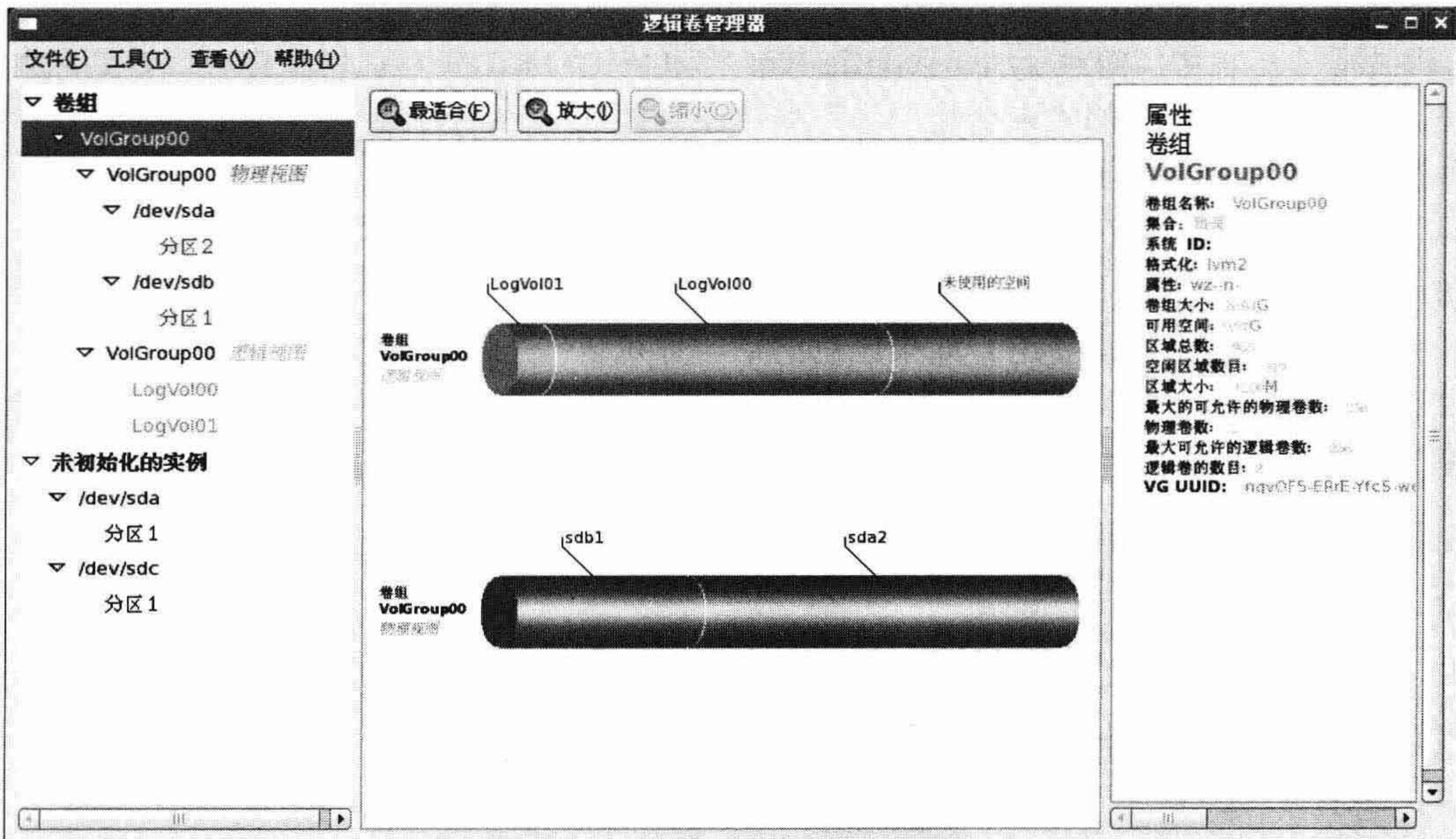


图 8-3 system-config-lvm 应用程序

你刚才添加了一个新磁盘，并用 fdisk 在其上创建了一个单独的整盘分区。我们要对它初始化，然后创建 1 个卷组和 1 个逻辑卷。该磁盘将在左侧面板的“未初始化的实例”菜单里出现，如图 8-4 所示。

你可以在右边面板看到新磁盘的参数。为了初始化新磁盘（即创建一个物理卷，就像使用 pvcreate 命令所做的那样），单击中间面板底端的“初始化实例”按钮。

图 8-5 显示即将擦除磁盘上/dev/sdd1 分区的警告。如果不想继续下去，单击“否”按钮；否则单击“是”按钮，继续前行。

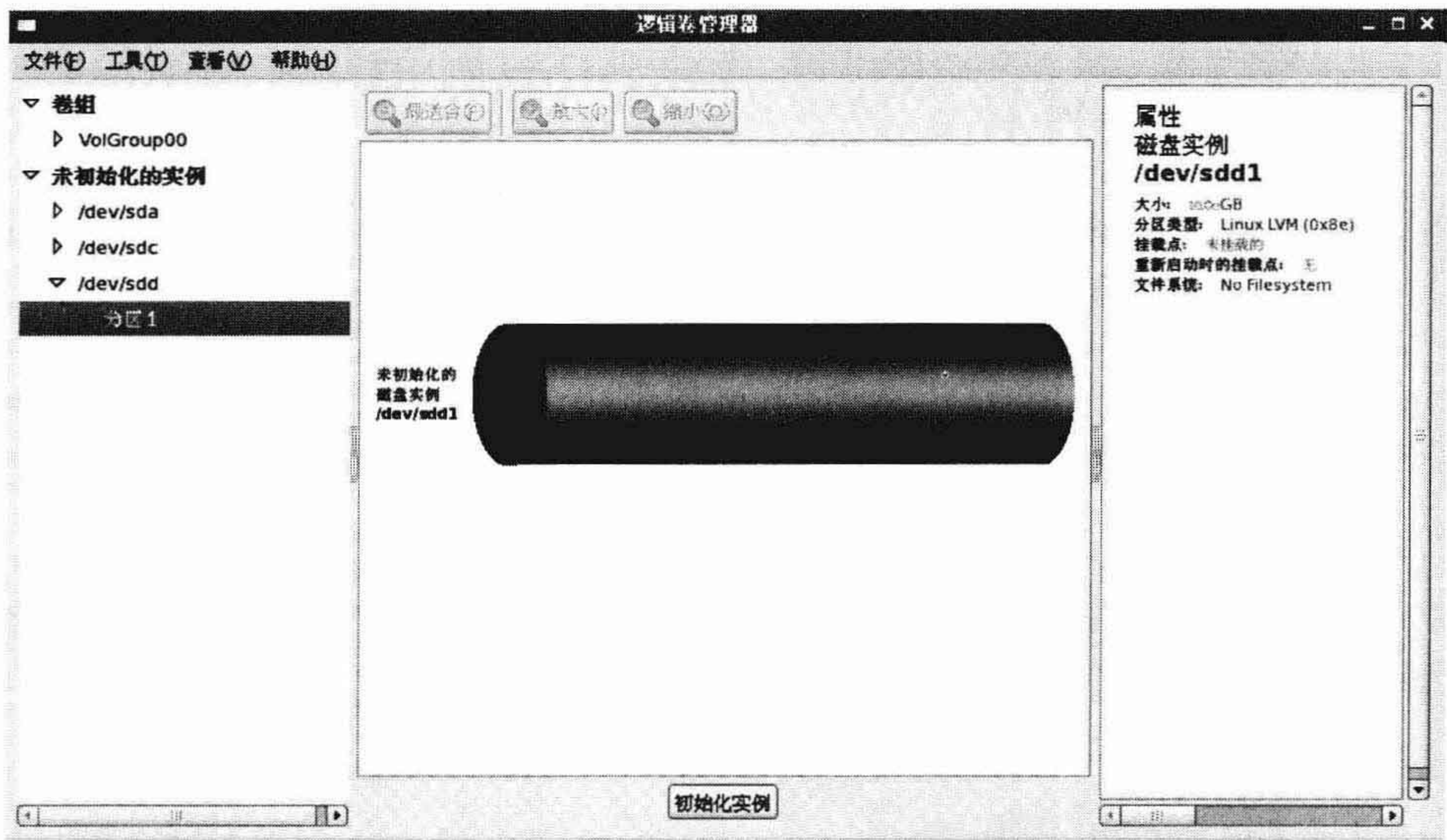


图 8-4 “未初始化的实例”菜单中的新磁盘

磁盘将被初始化，而且 system-config-lvm 将重载 LVM 的资料。现在你可以看到磁盘和分区已经移到左侧面板的“未分配的卷”菜单中。在右侧面板，你可以看到参数已经相应地改变了。如图 8-6 所示，在中间面板有三个选项供选择：“创建新的卷组”、“添加到已存在的卷组中”和“从 LVM 中移除卷”。

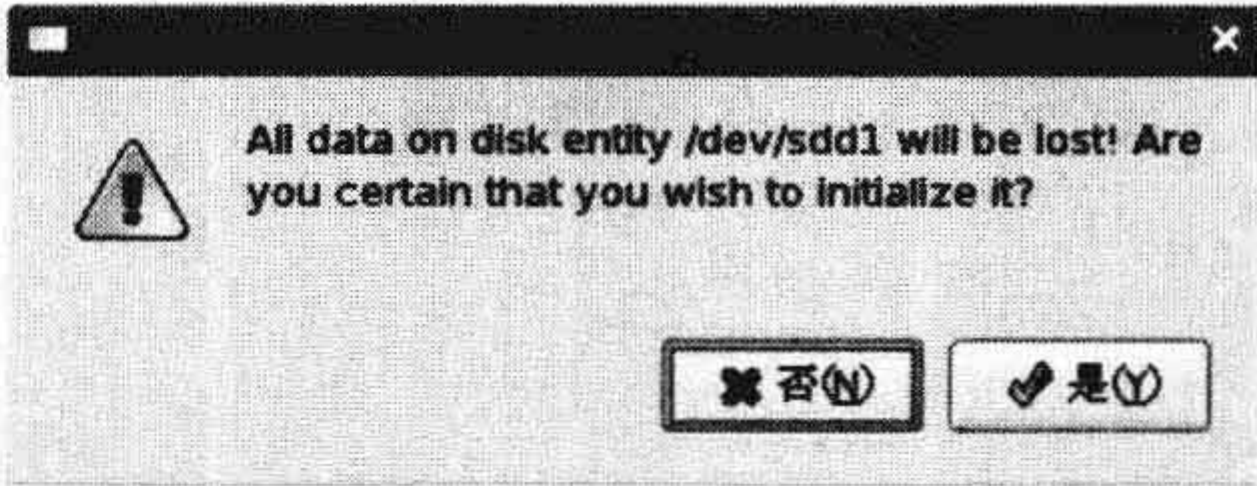


图 8-5 擦除分区前的警告

现在我们创建一个新卷组。如果愿意的话，你可以添加到一个已存在的卷组，那将为那个卷组提供额外的 10GB 可用空间。通过单击“创建新的卷组”创建这个新卷组。

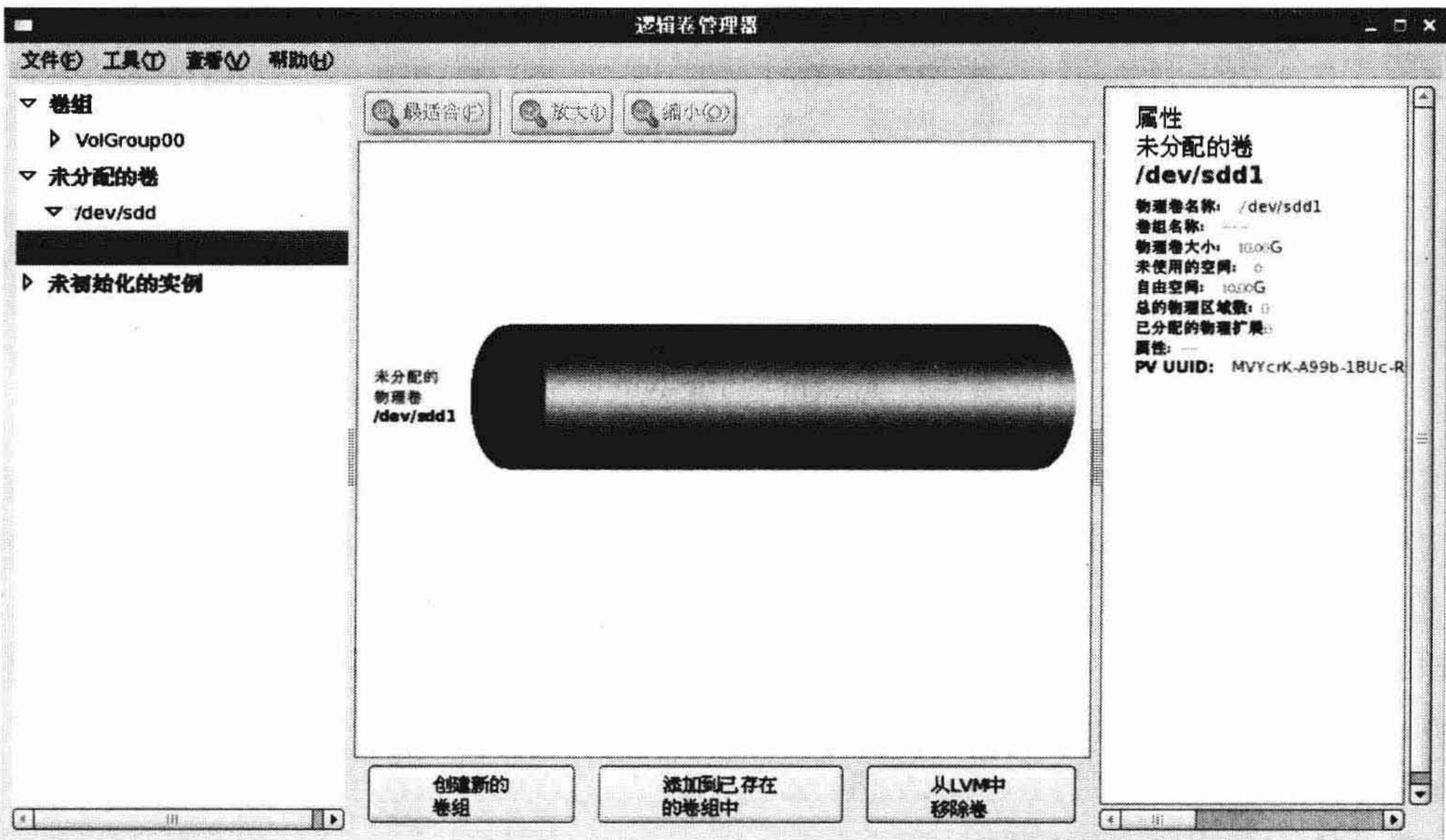


图 8-6 创建新卷组

在图 8-7 的屏幕截图中，它要求提供一些资料，如卷组名、磁盘容量和物理扩展容量。接受默认选项并给它取名 VolGroup01。这将创建一个可用空间为 10GB 的卷组。

从图 8-8 可以看到，磁盘和分区已经移动到左侧面板的“卷组”菜单下，现在已可以在它们上面创建逻辑卷。

接下来将只创建一个逻辑卷，操作方法是选中“VolGroup01 物理视图”，然后单击“创建新的逻辑卷”按钮，如图 8-8 所示。



图 8-7 指定卷组名

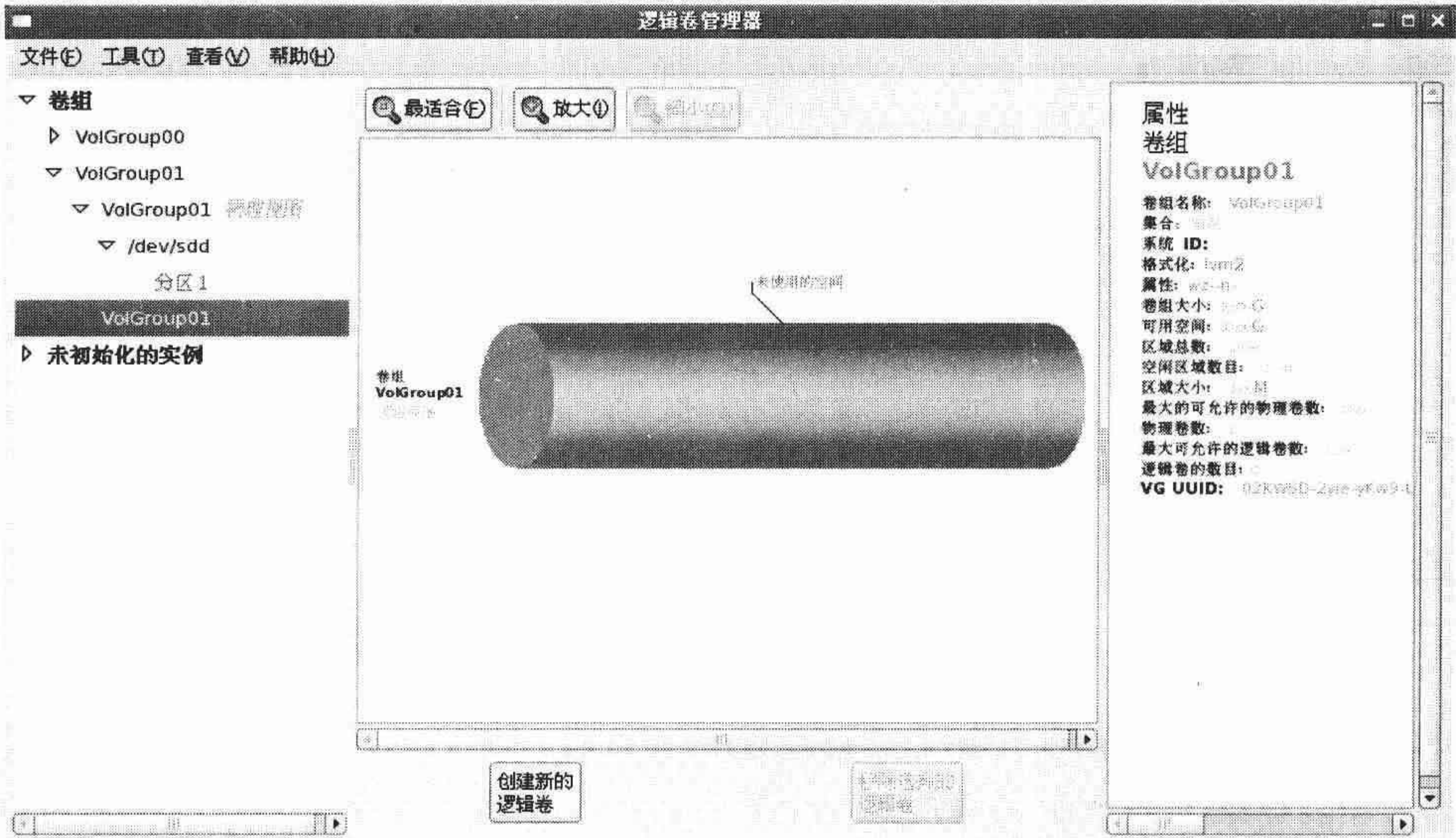


图 8-8 创建卷组 VolGroup01

在图 8-9 所示的“创建新的逻辑卷”对话框中，要求提供新逻辑卷的资料。我们给卷起名为 wwwData，使用全部可用的 10GB 容量，给它添加 ext3 文件系统，最后把它挂载在 /srv/www 目录下。

现在要回答是否希望确认创建挂载目录（见图 8-10）。这是新逻辑卷将要挂载的位置。选择“是”继续。system-config-lvm 现在将创建挂载目录并用 ext3 文件系统格式化此逻辑卷。

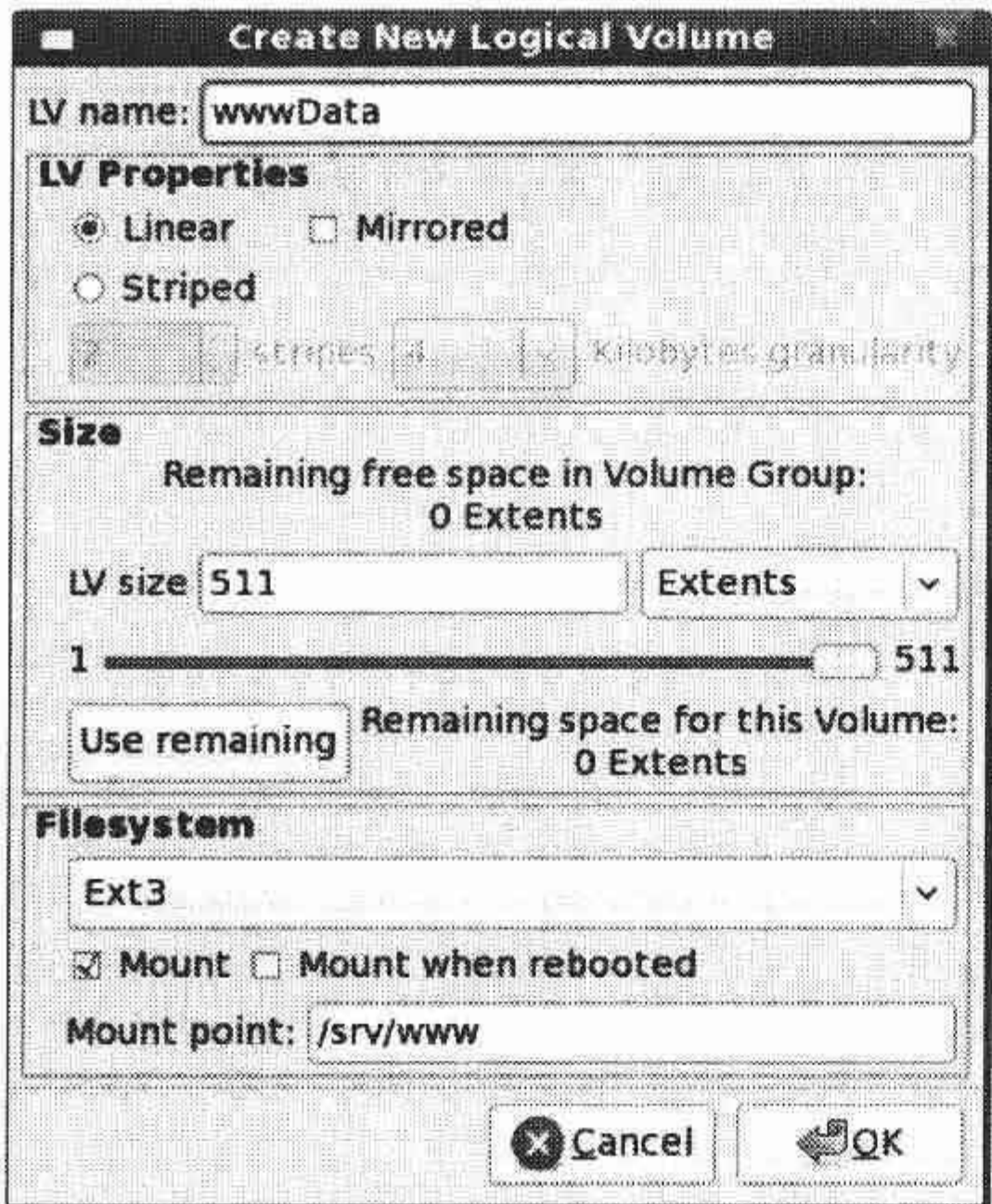


图 8-9 创建新的逻辑卷

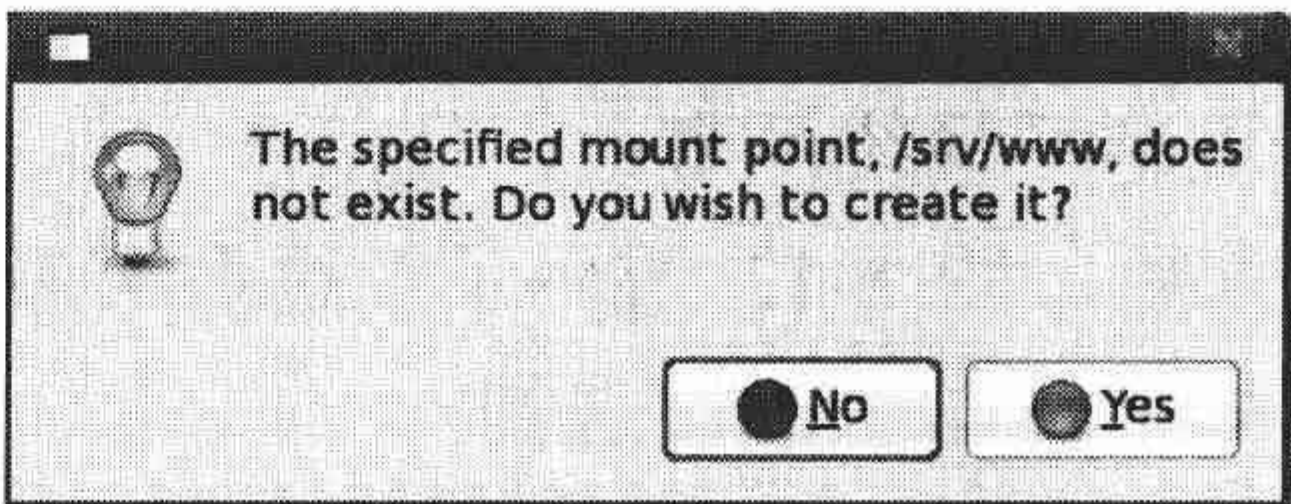


图 8-10 确认创建新逻辑卷

如图 8-11 所示，新逻辑卷 `wwwData` 已经创建完成。它是卷组 `VolGroup01` 的组成部分。后者由物理设备 `/dev/sdd` 的分区 1 组成。

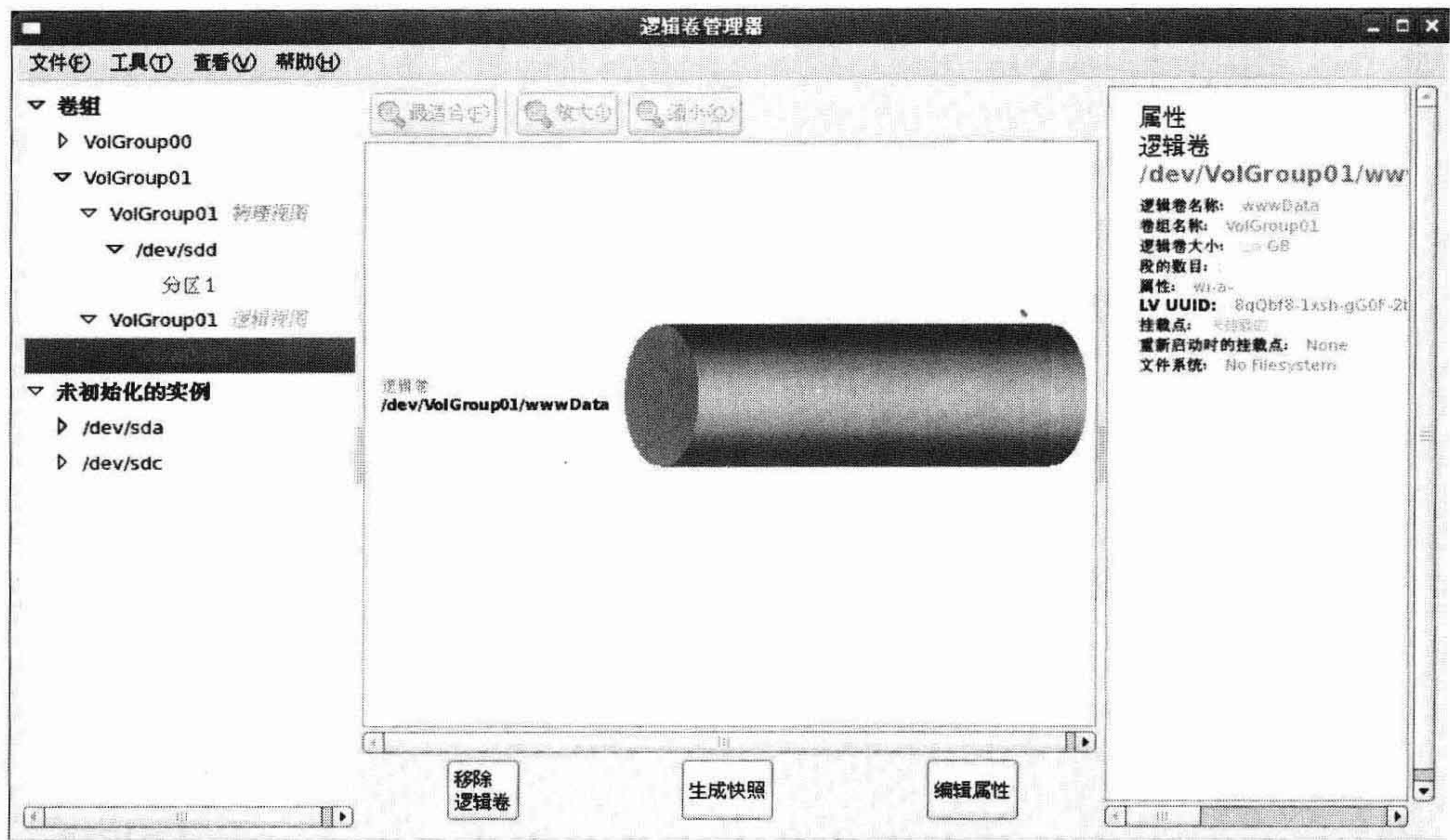


图 8-11 已创建完成的逻辑卷

你可以使用 `system-config-lvm` 工具移除逻辑卷或者创建它的快照。移除逻辑卷将毁坏其上的数据，并且在继续往下走之前得到一个警告。如果移除了一个逻辑卷，你就可以在那个卷组上创建一个或多个新逻辑卷。

你还可以用 `system-config-lvm` 移除卷组，但是必须确保它所拥有的所有逻辑卷先被移除。

创建逻辑卷的快照就是在某个时间点创建数据的一个副本。卷组要有足够的磁盘空间保存被拍快照的逻辑卷中的数据以及写入快照的任何新数据，而且保存时间要和此逻辑卷中的数据存在的时间一样长。这样你就可以离线移动特殊时间点的数据副本，从而可以在不影响逻辑卷的情况下做备份、检查或者某些种类的维护工作。

注：在第 13 章介绍备份与恢复时，我们将涉及更多有关 LVM 快照的内容。

8.5 故障恢复

如果主机遭受系统性事故，使某个文件系统处于不稳定的状态，系统就会在启动时自动尝试纠正此问题。图 8-12 显示了经过系统事故后系统是如何自动检查主机上的根文件系统以及如何重放 `/dev/sda1` 上的日志的。

这些自动恢复机制通常能工作，但有时修复工具也会发现它们不能自动解决的问题。

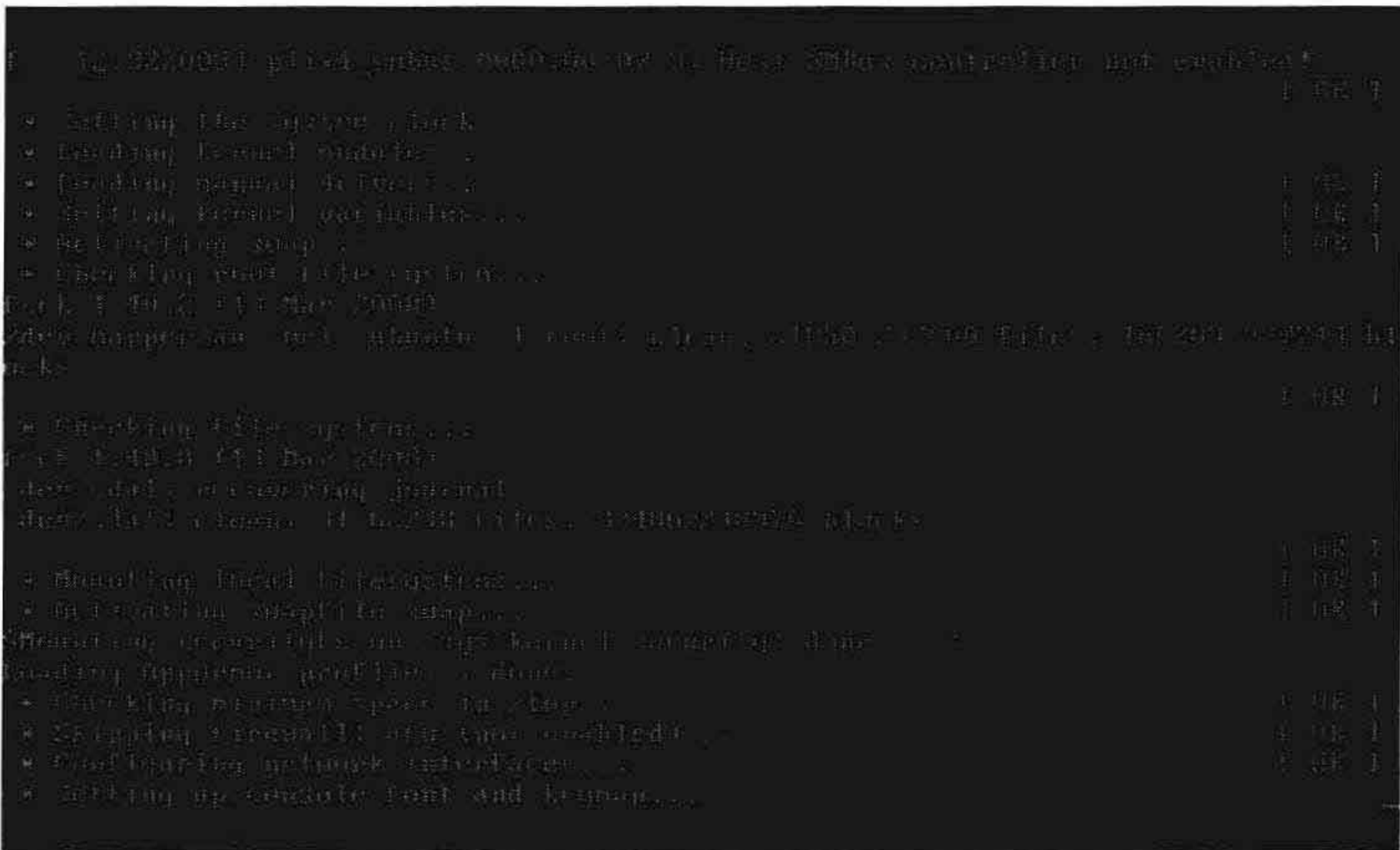


图 8-12 文件系统的自动恢复

如果根文件系统发生了这样的事情，而修复程序就存储在它里面，那么就需要从安装程序 CD 或 DVD 启动了。你可能可以从第 2 章回忆起来，安装程序 CD 或 DVD 包括一个挽救或恢复模式下启动的选项。这种模式引导进入一个从安装盘运行的最小系统，并进入一个 root shell。从这个 shell，你可以执行恢复文件系统所需的任何措施。

最简单的措施是运行适当的文件系统检查程序。表 8-6 详述了最可能用到的有关文件系统的系统检查与修复工具。

表 8-6 文件系统检查和修复工具	
文件系统	修复工具
ext2 和 ext3	e2fsck
XFS	xfs_repair
JFS	jfs_fsck

为了运行这些修复工具，你需要保证正在检查的文件系统不是为了写数据而挂载的。修复以读/写模式挂载的文件系统一定会毁坏数据，因为修复工具会在内核不知情的情况下直接修改文件系统。

为检查文件系统，你要给修复工具传递合适的设备节点名作为参数。

■注：当运行一个来自根文件系统的工具修复根文件系统本身时，请确保首先以只读的方式挂载该文件系统。

如果不是根文件系统有问题，系统会告知不能挂载该文件系统并继续启动，如图 8-13 所示。我们也注意到/dev/sda1 (/boot) 每挂载两次都会被自动检查一次。该文件系统早先并没有用 tune2fs -c 0 做修改。

当启动完成时，你可以登入系统并运行修复工具。列表 8-21 显示了对早前创建的 ext3 文件系统的检修过程。我们传递-p 选项给 e2fsck，让它自动修复那些无需用户干预就可解决的问题。如果存在其他问题，我们一般对任何问题都回答 Y 以修复它们。在列表 8-21 中，正在检查一个 ext3 文件系统。

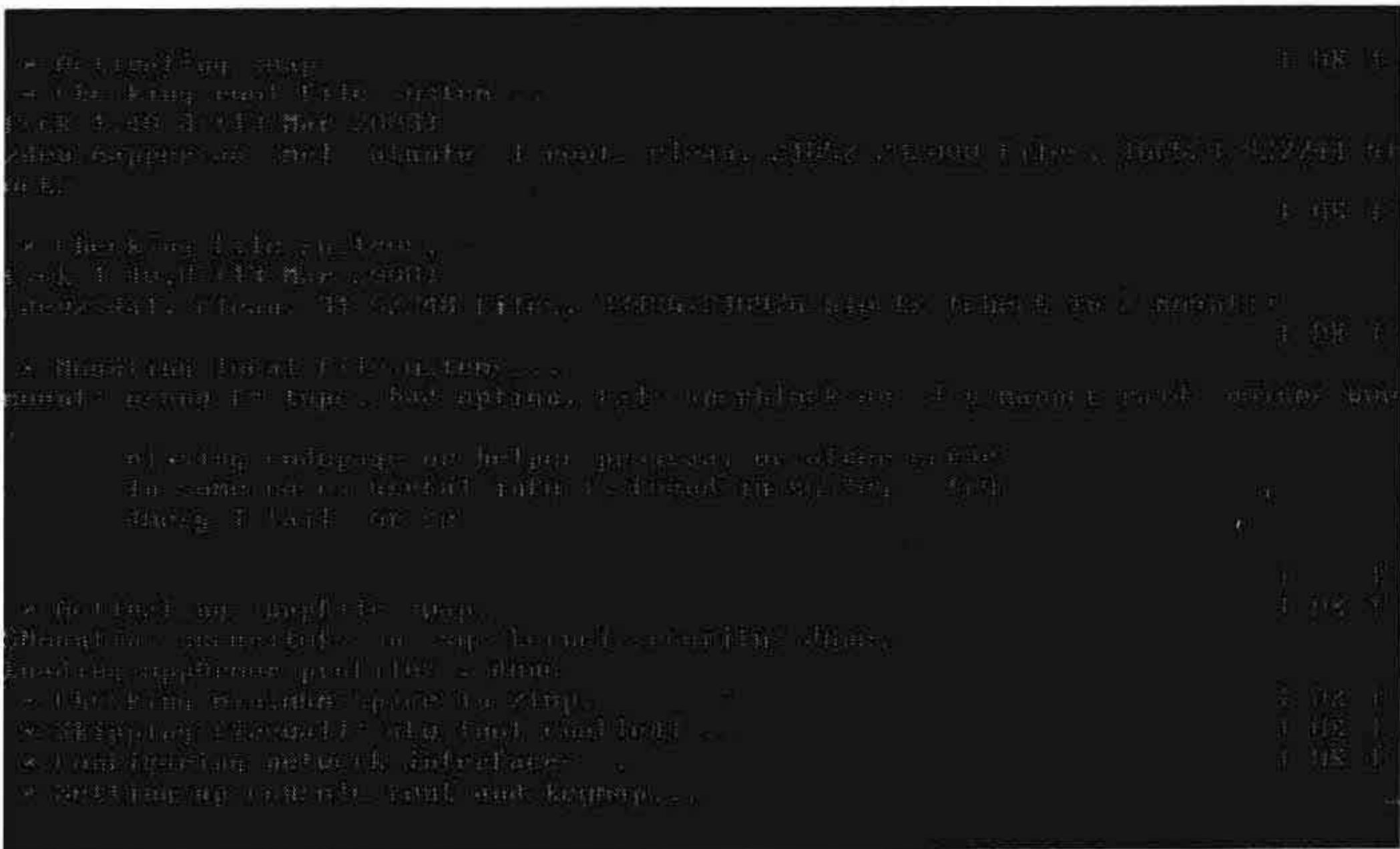


图 8-13 文件系统问题阻止了自动挂载

列表 8-21 修复 ext3 文件系统

```
# e2fsck -p /dev/raid-volume/www
e2fsck 1.40.8 (13-Mar-2008)
/dev/raid-volume/www contains a file system with errors, check forced.
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
Block bitmap differences: -(2--256) -520 -(32770--33024)
Fix<y>? yes

/dev/raid-volume/www: ***** FILE SYSTEM WAS MODIFIED *****
```

e2fsck 在运行中发现了可自行解决的问题并修复了它们。

有时与 ext2 或 ext3 文件系统有关的超级块有问题，这意味着文件系统检查工具不能定位修复文件系统所需要的文件系统元数据。为此，这些文件系统还保存了备用超级块。你可以用 dumpe2fs 工具显示它们在磁盘的位置，然后用 -b 选项为 e2fsck 指定此位置，如列表 8-22 所示。

列表 8-22 查找备用超级块

```
# dumpe2fs /dev/raid-volume/www | grep Backup
dumpe2fs 1.40.8 (13-Mar-2008)
Backup superblock at 32768, Group descriptors at 32769-32769
Backup superblock at 98304, Group descriptors at 98305-98305
Backup superblock at 163840, Group descriptors at 163841-163841
Backup superblock at 229376, Group descriptors at 229377-229377
Backup superblock at 294912, Group descriptors at 294913-294913
```

一旦知道了备用超级块的位置，你就可以试着运行 e2fsck 并用 -b 选项指定备用超级块的位置。

```
# e2fsck -b 32768 -p /dev/raid-volume/www
e2fsck 1.40.8 (13-Mar-2008)
Pass 1: Checking inodes, blocks, and sizes
```



```

Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/raid-volume/www: 11/131072 files (0.0% non-contiguous), 36947/524288 blocks

```

e2fsck 命令现在就可以运行并修复文件系统了。

如果有问题，并且在索引节点里发现了 **e2fsck** 不能处置的文件数据，这些数据将被存储在正被检查的文件系统中 **lost+found** 目录下的文件里。文件名是与此数据相关联的索引节点号。尽管这并没有表明哪些文件已经不在它们应该在的地方了，但是我们可以手动检查这些编了号的文件，并对那些可能丢失的数据产生一些感觉。

如果大量的数据被损坏，几百个文件在 **lost+found** 目录里，那么你可能最好从备份中恢复数据。

8.5.1 引导程序的问题

如果硬盘出了问题，系统可能随之不能启动，因为启动扇区损坏了。在这种情况下，引导程序不能启动，你在屏幕上将看到奇怪的错误，就像在第 5 章所显示的那样。

这些问题可通过重新安装引导程序来解决。第一步是在挽救或恢复模式里启动安装程序 CD 或 DVD，并从 **root shell** 运行文件系统检查，如刚才所示。当文件系统已被修复时，你就可以挂载它了。

```

# e2fsck -p /dev/sda1
# mount -t ext3 /dev/sda1 /mnt

```

你现在可以通过 **chroot**（修改 **root**）命令在已挂载的分区上登入 Linux 安装程序，这样可以为 **shell** 指定另一个系统 **root**。通过为已加载的根文件系统设置新 **root**，你就可以在正试图恢复的系统上得到一个 **root shell**。

运行 **chroot** 之前，你要确保可通过动态的 **/dev** 与 **/proc** 文件系统访问设备文件和内核接口。你可以把现存的文件系统挂载到根文件系统，并在其上通过绑定挂载运行 **chroot**。“绑定挂载”（**bind mount**）是一种无需符号链接就可使某个目录在系统多处使用的方式。在本例中，不能使用符号链接，因为它们会指向 **chroot** 外部的目标，而在 **chroot** 内部不能访问到。

```

# mount -o bind /dev /mnt/dev
# mount -o bind /proc /mnt/proc

```

当完成绑定挂载后，你可以 **chroot** 已挂载的根文件系统，并在那里运行一个 **Bash shell**。提示符看起来会有点奇怪，因为 **Bash** 启动脚本不会运行。

```

# chroot /mnt /bin/bash
bash-2 #

```

既然有了一个 **shell**，你就可以运行命令了，在第一个磁盘上重新安装引导程序。

```

bash-2 # /usr/bin/grub-install /dev/sda

```

完成之后，从 **chroot** 退出，并按倒序卸载文件系统。如果不卸载它们，已修复的根文件

系统就不能卸载，下次启动时就会又被强制进行一次检查。

```
bash-2 # exit
# umount /mnt/proc
# umount /mnt/dev
# umount /mnt
```

你现在可以使用 **Ctrl+Alt+Delete**、**reboot** 或者 **shutdown -r now** 命令重启主机。移除安装程序媒介，这样主机就从硬盘启动。

8.5.2 磁盘故障

这些硬盘中迟早会有一个出现故障，但是如果使用 **RAID**，这将不再是问题，不过需要在下一个磁盘出现故障前采取行动以保证阵列能够重新完整地工作。正如在本章早前看到的那样，你可以保持某个磁盘为备份盘，这样就使这个过程部分自动化了。

为了模拟磁盘故障，我们使用 **mdadm** 告诉内核/dev/sdb1 出现了故障，如列表 8-23 所示。

列表 8-23 RAID 阵列测试

```
$ sudo mdadm --manage /dev/md0 --fail /dev/sdb1
mdadm: set /dev/sdb1 faulty in /dev/md0
```

此时，**RAID** 子系统知道设备出现了故障，这与 **RAID** 驱动器检测到阵列中某一设备出现错误时的结果是一样的。让我们再看看/dev/mdstat 文件。

```
$ cat /proc/mdstat
Personalities : [linear] [multipath] [raid0] [raid1] [raid6] [raid5] [raid4]
               [raid10]
md0 : active raid5 sdf1[3] sdd1[2] sde1[4] sdc1[1] sdb1[5](F)
      25157376 blocks level 5, 64k chunk, algorithm 2 [4/3] [_UUU]
      [=====>.....] recovery = 38.6% (3242368/8385792) finish=0.7min
      speed=120087K/sec

unused devices: <none>
```

sdb1 磁盘此时作为有故障的设备被标记为 (F)，而备用磁盘 **sde1** 正在更新。你可以检查系统日志，以确保 **RAID** 监控程序已经了解这些改动并做了适当的反应。在 **Red Hat** 上，系统日志是/var/log/messages。而在 **Ubuntu** 上，它是/var/log/syslog。

```
Dec 31 12:53:52 au-mel-ubuntu-1 kernel: [68789.499377] raid5: Disk failure on
sdb1, disabling device. Operation continuing on 3 devices
Dec 31 12:53:52 au-mel-ubuntu-1 kernel: [68789.532848] RAID5 conf printout:
Dec 31 12:53:52 au-mel-ubuntu-1 kernel: [68789.532857] --- rd:4 wd:3
Dec 31 12:53:52 au-mel-ubuntu-1 kernel: [68789.532863] disk 0, o:0, dev:sdb1
Dec 31 12:53:52 au-mel-ubuntu-1 kernel: [68789.532866] disk 1, o:1, dev:sdc1
Dec 31 12:53:52 au-mel-ubuntu-1 kernel: [68789.532868] disk 2, o:1, dev:sdd1
Dec 31 12:53:52 au-mel-ubuntu-1 kernel: [68789.532871] disk 3, o:1, dev:sdf1
Dec 31 12:53:52 au-mel-ubuntu-1 kernel: [68789.561038] RAID5 conf printout:
Dec 31 12:53:52 au-mel-ubuntu-1 kernel: [68789.561044] --- rd:4 wd:3
Dec 31 12:53:52 au-mel-ubuntu-1 kernel: [68789.561054] disk 1, o:1, dev:sdc1
Dec 31 12:53:52 au-mel-ubuntu-1 kernel: [68789.561057] disk 2, o:1, dev:sdd1
Dec 31 12:53:52 au-mel-ubuntu-1 kernel: [68789.561059] disk 3, o:1, dev:sdf1
```



```

Dec 31 12:53:52 au-mel-ubuntu-1 kernel: [68789.561168] RAID5 conf printout:
Dec 31 12:53:52 au-mel-ubuntu-1 kernel: [68789.561170] --- rd:4 wd:3
Dec 31 12:53:52 au-mel-ubuntu-1 kernel: [68789.561173] disk 0, o:1, dev:sde1
Dec 31 12:53:52 au-mel-ubuntu-1 kernel: [68789.561175] disk 1, o:1, dev:sdc1
Dec 31 12:53:52 au-mel-ubuntu-1 kernel: [68789.561177] disk 2, o:1, dev:sdd1
Dec 31 12:53:52 au-mel-ubuntu-1 kernel: [68789.561180] disk 3, o:1, dev:sdf1
Dec 31 12:53:52 au-mel-ubuntu-1 kernel: [68789.562910] md: recovery of RAID
    array md0
Dec 31 12:53:52 au-mel-ubuntu-1 kernel: [68789.562944] md: minimum
    _guaranteed_ speed: 1000 KB/sec/disk.
Dec 31 12:53:52 au-mel-ubuntu-1 kernel: [68789.562948] md: using maximum
    available idle IO bandwidth (but not more than 200000 KB/sec) for recovery.
Dec 31 12:53:52 au-mel-ubuntu-1 kernel: [68789.562953] md: using 128k
    window, over a total of 8385792 blocks.
Dec 31 12:53:52 au-mel-ubuntu-1 mdadm: Fail event detected on md device
    /dev/md0, component device /dev/sdb1
Dec 31 12:53:52 au-mel-ubuntu-1 mdadm: RebuildStarted event detected on
    md device /dev/md0
Dec 31 12:53:52 au-mel-ubuntu-1 mdadm: SpareActive event detected on
    md device /dev/md0, component device /dev/sdb1
Dec 31 12:53:52 au-mel-ubuntu-1 postfix/pickup[6687]: 5978A4550:
    uid=0 from=<root>
Dec 31 12:53:52 au-mel-ubuntu-1 postfix/cleanup[8457]: 5978A4550:
    message-id=<20081231015352.5978A4550@au-mel-ubuntu-1.example.com>
Dec 31 12:53:52 au-mel-ubuntu-1 postfix/qmgr[5280]: 5978A4550:
    from=<root@au-mel-ubuntu-1.example.com>, size=1023, nrcpt=1 (queue active)
Dec 31 12:53:53 au-mel-ubuntu-1 postfix/local[8459]: 5978A4550
    to=<root@au-mel-ubuntu-1.example.com>, orig_to=<root>, relay=local, delay=1.1
    delays=0.41/0.37/0/0.29, dsn=2.0.0, status=sent (delivered to command:
    procmail -a "$EXTENSION")
Dec 31 12:53:53 au-mel-ubuntu-1 postfix/qmgr[5280]: 5978A4550:
    removed
Dec 31 12:54:52 au-mel-ubuntu-1 mdadm: Rebuild80 event detected on
    md device /dev/md0
Dec 31 12:55:01 au-mel-ubuntu-1 kernel: [68858.893259] md: md0: recovery done.
Dec 31 12:55:01 au-mel-ubuntu-1 kernel: [68858.985655] RAID5 conf printout:
Dec 31 12:55:01 au-mel-ubuntu-1 kernel: [68858.985661] --- rd:4 wd:4
Dec 31 12:55:01 au-mel-ubuntu-1 kernel: [68858.985665] disk 0, o:1, dev:sde1
Dec 31 12:55:01 au-mel-ubuntu-1 kernel: [68858.985668] disk 1, o:1, dev:sdc1
Dec 31 12:55:01 au-mel-ubuntu-1 kernel: [68858.985670] disk 2, o:1, dev:sdd1
Dec 31 12:55:01 au-mel-ubuntu-1 kernel: [68858.985672] disk 3, o:1, dev:sdf1
Dec 31 12:55:01 au-mel-ubuntu-1 mdadm: RebuildFinished event detected
    on md device /dev/md0
Dec 31 12:55:01 au-mel-ubuntu-1 mdadm: SpareActive event detected
    on md device /dev/md0, component device /dev/sde1

```

好极了！监控程序了解了这次故障，激活了备用设备，启动了数据复原，发送了一封 E-mail，并完成了阵列的重建。RAID 系统已经保存了我们的数据，并且阵列仍然是完整的。对我们来说，剩下要做的就是从阵列中移除有故障的磁盘，然后把它替换下来。首先，你要调用 mdadm 把有故障的磁盘从 RAID 阵列中移除。

```

$ sudo mdadm --manage /dev/md0 --remove /dev/sdb1
mdadm: hot removed /dev/sdb1

```


接下来的措施取决于硬盘控制器。如果它支持热插拔驱动器，就可以拔掉坏磁盘，用一个新的替代它。如果不支持，则必须关掉主机，自然地替换此驱动器。

当安装完新驱动器并启动了主机，你将需要像对阵列中其他磁盘那样对新磁盘分区。新磁盘很可能与它替换下的磁盘有相同的设备节点名。当分区完成后，你可通过 `mdadm` 向阵列添加新分区，如列表 8-24 所示。

列表 8-24 向 RAID 阵列添加新设备

```
$ sudo mdadm --manage /dev/md0 --add /dev/sdb1
mdadm: added /dev/sdb1
$ cat /proc/mdstat
Personalities : [linear] [multipath] [raid0] [raid1] [raid6] [raid5] [raid4]
               [raid10]
md0 : active raid5 sdb1[4] (S) sdf1[3] sdd1[2] sde1[0] sdc1[1]
      25157376 blocks level 5, 64k chunk, algorithm 2 [4/4] [UUUU]

unused devices: <none>
```

新磁盘以备用盘的角色加入阵列，它将在下一次磁盘故障时接任。

可能会有某种问题阻止 RAID 阵列自动启动。如果在挽救模式下从 CD 或者 DVD 启动主机，阵列就可能不会被检测到，也不会启动。如果发生了这种情况，你可以手动装配阵列。为了从现存的组件重新组建 RAID 阵列，你可使用装配模式下的 `mdadm` 程序。它将尝试从指定的组件装配阵列。更多相关信息可到 `mdadm` 手册页面查看。

8.6 小 结

在本章，你学习了如何在 Linux 上管理存储，以及如何安全又灵活地利用 RAID 和 LVM 存储数据。现在你可以创建分区和文件系统，也可以手动或在启动时自动挂载和卸载文件系统。你还学习了如何设置软 RAID 和 LVM，以及调整卷和文件系统的大小。在磁盘出现故障的情况下，你知道怎样在 RAID 阵列中移除和添加组件，并能够修复文件系统错误。

第 9 章将讲述怎样配置如 SSH 服务器这样的基础架构服务，探讨怎样在主机上使用 NTP 管理时间，并介绍 DNS 和 DHCP。

第二部分

让 Linux 为你工作

第 9 章 基础架构服务：NTP、DNS、DHCP 和 SSH

在前面几章，我们安装了主机，并开始了解它的操作方式。然后我们学习了如何添加和配置存储器硬件。现在应该看看如何让软件服务于你。本章将涉及帮助管理这些网络基本构件的基础架构服务。

首先我们将介绍如何让系统时间保持同步。这很重要，因为很多应用程序依赖主机上正确的时间。在此过程中介绍网络时间协议（Network Time Protocol, NTP）。

我们还将涉及域名系统（Domain Name System, DNS）。它是胶合剂，通过让主机找到彼此而使如 Internet 这样的网络工作。本章将介绍 DNS 组件以及如何设置和管理 DNS 服务器。

然后我们将讲述动态主机配置协议（DHCP）。它用来给主机分配地址和网络配置。使用 DHCP 意味着不必为网络中的客户机配置单机网络设置；它可以自动提供。你将学习如何使用 DHCP，如何设置地址的分配以及给主机传送网络配置信息。

■注：第 19 章会介绍其他自动配置主机的方式。

最后我们将详细阐述安全 shell (SSH) 服务，并说明如何轻松访问主机以及如何使用 SSH 在主机之间传送文件。

9.1 网络时间协议

我们首先说明如何保持主机上所有系统时钟的同步。尽管这可能看起来是一个微不足道的问题，但是系统时钟的匹配意味着日志输入项具有一致的时间戳。反过来说，这意味着可以轻易地关联来自不同主机的日志输入项，如果有这种需要的话。同步的系统时钟也是后面所启用功能的先决条件。不能简单地依赖主机主板上的板上时钟，因为它们的质量差别很大，有些时钟一天走的时间可以和同步时间差好几分钟。

时间服务由一个叫作网络时间协议（Network Time Protocol, NTP）的服务提供。NTP 服务器为连接到它们的客户机提供同步服务，并且它们自己也和上游的时间服务器同步。在

这个模型中这些层叫作 **strata**，最高级别为 **stratum 0**，它们由专用的时间硬件组成，如原子钟或卫星接收器。连接到这些 **stratum 0** 时间资源的服务器叫作 **stratum 1** 服务器。通过 **stratum 1** 服务器同步的服务器是 **stratum 2** 服务器等。

■注：在 http://www.akadia.com/services/ntp_synchronize.html 上可以阅读更多有关 NTP strata 的信息。

你可以用两种方式使用 NTP 服务器。一种方式是运行一个叫作 **ntpd** 的客户机实用程序同步一次系统时钟。另一种方式是运行一个 NTP 服务，无论系统时钟何时偏离真实时间太远，它都会自动同步。实际上，很多系统两种方式都用。如果系统时钟和原子钟时间相差太多，它就可以花费一点时间让系统和上游的时间服务器同步。为了克服这个问题，它调用 **ntpd** 实用程序，时钟在 NTP 服务开启之前就得以同步。

让我们首先看一看 **ntpd** 实用程序。在 Red Hat 和 Ubuntu 上它都是由 **ntpd** 软件包提供。为了更新系统时间，我们用上游服务器地址作为唯一的命令行参数来运行该程序。为了能够更新系统时钟，我们需要以 **root** 用户身份运行它。

```
$ sudo ntpdate pool.ntp.org
18 Jan 12:56:25 ntpdate[8695]: adjust time server 203.19.252.1 offset 0.089414 sec
```

ntpd 程序连接到 **pool.ntp.org** 服务器中的一个并把我们的系统时间调校到 1/900sec 内。一般在 Ubuntu 上，当某个网络接口通过 DHCP 客户端连上网的时候就会运行此程序。为了确保系统时钟保持同步，你可以在 **/etc/crontab** 里添加一个输入项，让它每两个小时运行一次 **ntpd**。把标准输入和标准输出重定向到 **/dev/null**，这样你就不会接收到两个小时一次的电子邮件了。

```
0 */2 * * * root /usr/sbin/ntpdate pool.ntp.org > /dev/null 2>&1
```

不过，你需在每一台主机上安装并维护这样一个 **crontab**，甚至到那时，根据硬件品质的不同，系统时钟可能在两个小时的时间段里偏离很多。通过在主机上安装并运行一个 NTP 服务器，你可以确保系统时钟无论何时试图偏离同步时间，都可以得到调整。这使主机保持同步，还可以使用它同步网络上的其他主机。

NTP 服务器和一些相关联的实用程序由 **ntp** 软件包提供。你需要在 Red Hat 上通过 **yum install ntp** 或者在 Ubuntu 上通过 **sudo aptitude install ntp** 安装它。在启动时，**ntpd** 服务会从 **/etc/ntp.conf** 文件读取它的选项并监听 UDP 端口 123。如果查看这个配置文件，你可以看到它由两个主要的部分组成：第一个是真实时间源配置，第二个是授权配置。我们从列表 9-1 所示的报告和时间源配置开始。

列表 9-1 ntp.conf

```
# /etc/ntp.conf, configuration for ntpd; see ntp.conf(5) for help

driftfile /var/lib/ntp/ntp.drift

# Enable this if you want statistics to be logged.
#statsdir /var/log/ntpstats/
```



```
statistics loopstats peerstats clockstats
filegen loopstats file loopstats type day enable
filegen peerstats file peerstats type day enable
filegen clockstats file clockstats type day enable

# You do need to talk to an NTP server or two (or three).
server ntp.ubuntu.com
```

driftfile 指令为服务器提供一个存储本地系统时钟特性信息的地方。随着时间的流逝，它会使用这个信息在两次同步尝试之间更精确地报告时间，因为这个后台程序了解本地时钟的行为方式。

统计报告不是缺省打开的，因为 **statsdir** 选项没有打开。不过，如果去掉那一行的注释符号，接下来的指令 **statistics** 就会开启 **loopstats**、**peerstats** 和 **clockstats** 报告，并把它们存放到 **/var/log/ntpstats** 里的文件中。

loopstats 搜集由 **ntpd** 服务器向本地时钟所做的更新信息。**peerstats** 记录有关所有对等体的信息——上游服务器和使用你的服务器进行同步的客户机。最后 **clockstats** 往日志文件里写入本地时钟相关的统计信息。

filegen 指令告诉这个后台程序希望这些统计信息写入哪个文件以及这个文件多久改动一次。在我们的例子中，由于 **type day** 指令，这些文件每天都会被创建新的版本。

最后，**server** 选项告诉 **ntpd** 要使用哪一个上游服务器来同步。为了确保主机保持同步，添加多个带不同服务器的服务器指令通常是个好主意。一会儿就将说明如何找到这些服务器。

首先快速看一看 **/etc/ntp.conf** 文件里接下来的部分，它定义了哪些主机可以访问 NTP 服务器。在 Red Hat 上，这部分列在文件顶部，如列表 9-2 所示。

列表 9-2 ntp.conf 里的访问控制

```
# By default, exchange time with everybody, but don't allow configuration.
restrict -4 default kod notrap nomodify nopeer noquery
restrict -6 default kod notrap nomodify nopeer noquery
# Local users may interrogate the ntp server more closely.
restrict 127.0.0.1
restrict ::1
```

restrict 关键词用来定义访问类别。因为使用了 -4 和 -6 参数，所以在这里 IPv4 和 IPv6 客户机被定义了相同的访问级别。

default 是一个匹配所有可能地址的通配符关键词。**kod** 用来发送特定的回应包，以减慢那些超过规定速度界限的客户机。因为还没有规定这样的界限，所以在这里没用到它。**notrap** 拒绝任何被发送的控制包，而 **nomodify** 禁止修改服务器时间的尝试。**nopeer** 确保服务器不使用一个正在连接的客户端作为上游 NTP 服务器。最后，**noquery** 阻止别人查询服务器的对等体和其他统计信息。

第二组 **restrict** 指令确保来自本地机器的连接可以询问和重配置 NTP 服务器。不过，所有这些指令都没有阻止客户机与 NTP 服务器进行同步。

■注：在 <http://www.cis.udel.edu/~mills/ntp/html/> 上可以找到更多关于 NTP 配置和访问控制的信息。

全球 NTP 服务器池

许多组织运行它们自己的时间服务器，并允许第三方访问它们。Microsoft 和 Apple 运行的时间服务器默认用于它们各自的操作系统，许多 Linux 厂商也这样做。

无论如何，当希望往自己的 `ntp.conf` 文件中添加另外的服务器时，你需要知道它们的地址。很幸运的是，有一个开源项目的目的就是为所有大陆提供本地 NTP 服务器池。这个项目叫作 `pool.ntp.org`，参与者是个人用户和允许第三方使用其服务器进行同步的组织。

该项目为各种服务器 `strata` 和地理位置提供基于 DNS 的组——例如，`1.pool.ntp.org` 由 `stratum 1` 服务器提供，`au.pool.ntp.org` 只包括位于澳大利亚的服务器，`us.pool.ntp.org` 由位于美国的服务器提供。通过添加选定的 `pool.ntp.org` 服务器，就确保总有最新和最近的服务器可同步。

■注：在 `http://www.pool.ntp.org/` 上可以了解该项目并加入这个服务器池。

我们在 `/etc/ntp.conf` 文件里向服务器部分添加一个 `stratum 1` 服务器和一个本地服务器，如列表 9-3 所示。

列表 9-3 定义另外的上游 NTP 服务器

```
server ntp.ubuntu.com
server 1.pool.ntp.org
server au.pool.ntp.org
```

添加好这些服务器之后，你就可以在 Ubuntu 上使用 `sudo invoke-rc.d ntp restart` 命令或者在 Red Hat 上使用 `sudo service ntpd restart` 命令重启 NTP 服务器。该服务器把所有状态更新写入系统记录器；你可以在 Ubuntu 的 `/var/log/syslog` 里或者在 Red Hat 的 `/var/log/messages` 里找到它们。列表 9-4 显示了一个开启之后与上游服务器同步过的服务器的输出。

列表 9-4 系统记录里的 `ntpd` 状态

```
Feb 1 17:00:47 au-mel-ubuntu-1 ntpd[26855]: ntpd 4.2.2p4@1.1585-o Mon
Jan 5 19:56:03 UTC 2009 (1)
Feb 1 17:00:47 au-mel-ubuntu-1 ntpd[26856]: precision = 1.000 usec
Feb 1 17:00:47 au-mel-ubuntu-1 ntpd[26856]: Listening on interface
wildcard, 0.0.0.0#123 Disabled
Feb 1 17:00:47 au-mel-ubuntu-1 ntpd[26856]: Listening on interface
wildcard, ::#123 Disabled
Feb 1 17:00:47 au-mel-ubuntu-1 ntpd[26856]: Listening on interface
lo, ::1#123 Enabled
Feb 1 17:00:47 au-mel-ubuntu-1 ntpd[26856]: Listening on interface
eth0, fe80::216:6cff:fe8e:d687#123 Enabled
Feb 1 17:00:47 au-mel-ubuntu-1 ntpd[26856]: Listening on interface
lo, 127.0.0.1#123 Enabled
Feb 1 17:00:47 au-mel-ubuntu-1 ntpd[26856]: Listening on interface
eth0, 192.168.0.1#123 Enabled
Feb 1 17:00:47 au-mel-ubuntu-1 ntpd[26856]: kernel time sync status 0040
Feb 1 17:00:47 au-mel-ubuntu-1 ntpd[26856]: frequency initialized -18.989
PPM from /var/lib/ntp/ntp.drift
```



```
Feb 1 17:01:02 au-mel-ubuntu-1 ntpd[26856]: synchronized to 202.60.65.243,
stratum 2
Feb 1 17:01:02 au-mel-ubuntu-1 ntpd[26856]: synchronized to 203.19.252.1,
stratum 2
Feb 1 17:01:02 au-mel-ubuntu-1 ntpd[26856]: kernel time sync enabled 0001
```

你还可以使用 `ntpd` 命令，通过从本地主机查询 NTP 服务器的方式验证主机已同步的事实，如列表 9-5 所示。我们使用 `-p` 选项列出所有连接过的对等体，使用 `-4` 选项把主机名解析为 IPv4 地址。

列表 9-5 列出连接过的对等体

```
$ ntpq -4 -p localhost
      remote           refid      st t when  poll reach delay  offset  jitter
=====
+europium.canoni 193.79.237.14  2 u    5   64  377 350.024 -59.414  1.069
+fel-1.mel-ii.bd 203.36.227.2   2 u   64   64  377  71.646 -48.217  1.993
*ns.tti.net.au 203.36.227.2   2 u    2   64  377  69.299 -58.937  1.630
```

你现在可以配置网络上其他任何主机，让它们使用堡垒主机作为上游 NTP 服务器并确保它们通过 `ntpd` 命令工作。

9.2 域名系统

在第 2 章建议对主机使用描述性的名字。不过，如果不提供把这些名字翻译成主机所配置地址的方法，你就不能使用这些名字远程访问这些主机。在大多数计算机系统中，这种域名到地址的映射关系被存储在一个“hosts”文件里，正如第 3 章所讲的那样。

可是，一旦你网络上的主机增长到不止几台时，保证这个文件的所有复制都保持同步将变成一件费劲的事，你可能就要考虑实现一个域名系统（Domain Name System，DNS）服务器。

注：在存在 DNS 之前，人们使用一个单独的文件 `hosts.txt`。这个文件由网络信息中心（Network Information Center，NIC）维护并通过 FTP 分发给连接到 ARPANET 的所有机器。

DNS 服务器维护地址到主机名（反之也一样）的映射列表，可以由其他主机或直接由用户使用各种实用工具查询。

9.2.1 根服务器

DNS 服务器要设法知道到哪个或哪些主机查询正确的地址。一个 `apple.com` 的 DNS 服务器不知道一个 `google.com` 主机，那么我们自己的 DNS 服务器如何知道去哪儿寻找？

整个 DNS 结构就像一个巨大的倒置树。一个域名里的每个句点就像树的一个分枝。随着从左到右阅读一个域名，每个句点都表示此树更低一级的拆分，并且更接近树根。这些级别被称为区域（zone），而且对于一个域所属的每个区域，都要做一次查询以找到那个区域的域

名服务器。然后反过来查询这些服务器中的某个服务器以获得下一级区域的 DNS 服务器。最低级别的区域——其他所有的区域都是其成员——叫作根区域。我们用单个句点表示这个区域。下一级别由顶级域（top-level domains, TLDs）组成，包括通用域（如 net、com、org 和 edu）以及国家代码（如 au、nz、uk 和 us）。图 9-1 显示了这个树结构的一小部分。

在写一个主机名或域名时，一般不写根区域的后缀句点，但是当涉及 DNS 服务器时，应该明确包含它，因为漏写会导致意外的结果。这些 TLD 的 DNS 信息存储在所谓的根服务器中。

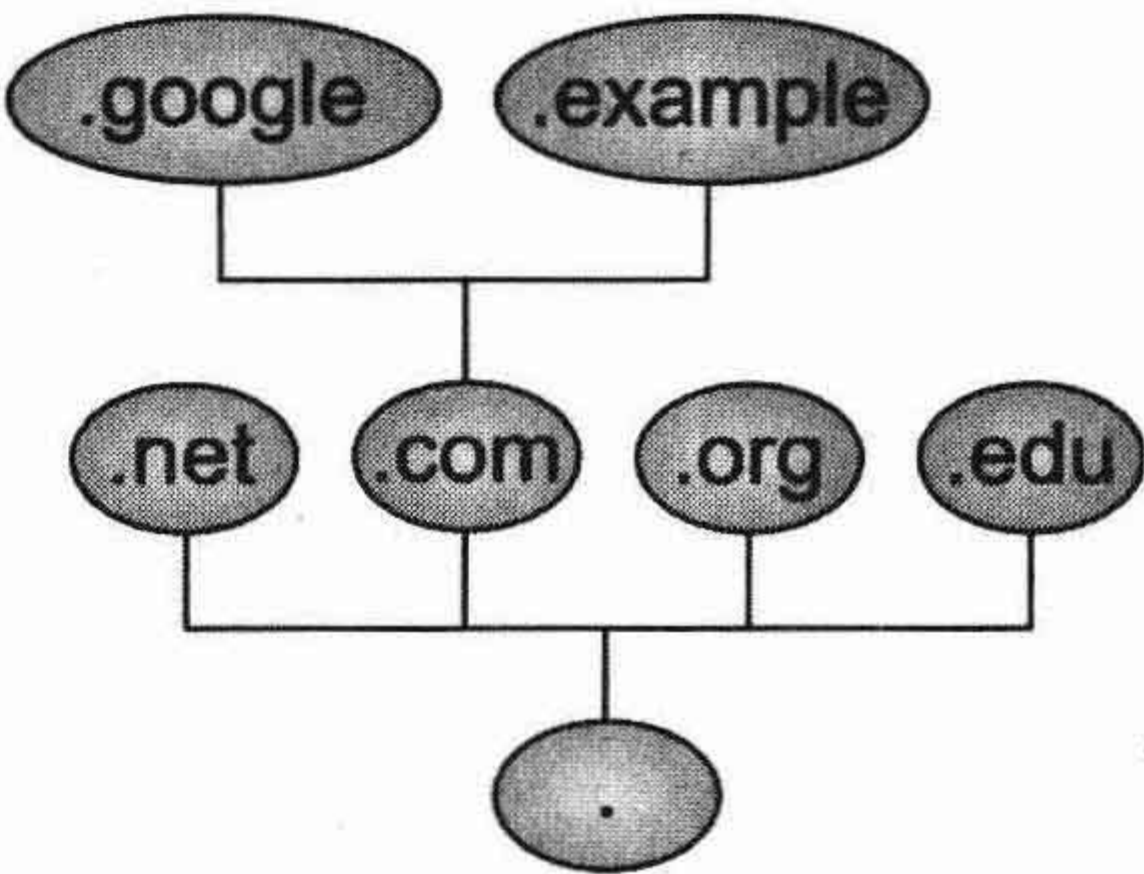


图 9-1 DNS 树结构

WHOIS

当一个组织或个人购买一个域时，DNS 根服务器要知道沿着树结构往下的哪些 DNS 服务器知道这个域。经管.com 域的组织是 Internet 域名和数字分配公司（Internet Corporation for Assigned Names and Numbers, ICANN），它管理登记机构。

从登记机构那里购买域时，你要指定这个域所委托的域名服务器。然后登记机构确保你的 DNS 服务器添加到正确的 TLD 区域，这样第三方就可以使用你的 DNS 服务器查找你域上的主机名了。

通过直接查询登记机构的数据库，你可以获得特定域的 DNS 服务器列表。为此所使用的工具是谁is，它有助于确保 DNS 委托的正确性。在 Red Hat 上它由 jwhois 软件包提供，在 Ubuntu 上它由 whois 软件包提供。安装此包，然后让我们看看 google.com 域的委托细节，如列表 9-6 所示。

列表 9-6 使用 whois 检查委托资料

```
$ whois google.com
[Querying whois. verisign-grs.com]
[whois.verisign-grs.com]
```

Whois Server Version 2.0

Domain names in the .com and .net domains can now be registered with many different competing registrars. Go to <http://www.internic.net> for detailed information.

```
Domain Name: GOOGLE.COM
Registrar: MARKMONITOR INC.
Whois Server: whois.markmonitor.com
Referral URL: http://www.markmonitor.com
Name Server: NS1.GOOGLE.COM
Name Server: NS2.GOOGLE.COM
Name Server: NS3.GOOGLE.COM
Name Server: NS4.GOOGLE.COM
Status: clientDeleteProhibited
Status: clientTransferProhibited
Status: clientUpdateProhibited
Status: serverDeleteProhibited
Status: serverTransferProhibited
Status: serverUpdateProhibited
Updated Date: 18-nov-2008
```



```
Creation Date: 15-sep-1997
Expiration Date: 14-sep-2011
>>> Last update of whois database: Sun, 01 Feb 2009 20:46:19 EST <<<
```

在这里 whois 连接到 VeriSign WHOIS 服务器并检索此域的一些信息，包括登记机构的名字，它所委托的域名服务器，创建者、修改者和到期日期。

一些登记机构还通过 whois 提供域拥有者的联系资料。当选择登记机构来购买域时要谨记这一点，因为这对于垃圾邮件发送者来说是一种相对方便的搜集电子邮件地址的方式。

■注：大多数登记机构只允许一个特定地址每天对它们的数据库进行有限次的查询，以阻止收集地址的行为。

9.2.2 查询域名服务器

你已经在使用 DNS 服务器或由 Internet 服务提供商所运行的服务器查询 Internet 上的主机地址。无论何时想做网站搜索，键入 `www.google.com` 总比记住 `74.125.19.147` 方便得多。这些 DNS 服务器的地址存储在 `/etc/resolv.conf` 文件中。列表 9-7 中包含了我们的 DNS 服务器；你的 DNS 服务器当然会不一样。

列表 9-7 `/etc/resolv.conf`

```
$ cat /etc/resolv.conf
search example.com
nameserver 192.168.1.1
nameserver 192.168.1.254
```

当访问一个网站或通过 SSH 连接到某个主机时，相应的应用程序使用这些 DNS 服务器执行一次主机查询。这些应用程序使用一个系统库，此系统库首先检查 `/etc/hosts` 文件，然后只有需要时才查询某个域名服务器。

1. host 命令

你也可以人工查询 DNS 服务器。与 DNS 相关的工具在 Red Hat 上由 `bind-utils` 软件包提供，在 Ubuntu 上由 `dnsutils` 软件包提供，那么安装它们。直接的主机或地址查找可以通过 `host` 实用程序完成。

■注：你可能习惯使用遭到反对的 `nslookup` 实用程序。`host` 命令是它的替代品。

我们传递想查询的主机名或地址，要查询的 DNS 服务器为可选项，如列表 9-8 所示。如果不写 DNS 服务器，程序会使用在 `/etc/resolv.conf` 里定义的一个服务器。

列表 9-8 用 `host` 查询一个 DNS 服务器

```
$ host www.google.com 192.168.1.1
www.google.com is an alias for www.l.google.com.
www.l.google.com has address 74.125.47.99
```



```
www.l.google.com has address 74.125.47.147
www.l.google.com has address 74.125.47.104
www.l.google.com has address 74.125.47.103
```

在列表 9-8 中，我们要求运行在 192.168.1.1 上的 DNS 服务器查询 `www.google.com` 的地址，它判定这实际上是一个别名，并依次指向 4 个不同的地址。
反过来，你也可以要求服务器查询这些地址中任何一个主机名。

```
$ host 74.125.47.99
99.47.125.74.in-addr.arpa domain name pointer yw-in-f99.google.com.
```

这个主机名与 `www.l.google.com` 不同。让我们用这个新主机名做一次查询，看看它是否有同一个地址。

```
$ host yw-in-f99.google.com
yw-in-f99.google.com has address 74.125.47.99
```

它确实有同样的地址，由此看来，这个地址分配了两个名字。

2. dig 命令

`host` 虽然有用，但是它提供的信息通常不足以帮助解决可能存在的 DNS 问题，特别当运行自己的 DNS 服务器时更是如此。`dig` 是一个更灵活的实用程序，它也由 `bind-utils` 或 `dnsutils` 软件包提供。

在它的最基本层面上，`dig` 也做基于域名或地址的查询，但每次查询都提供额外的信息。让我们做一下在列表 9-8 中做过的同样的查询，不过改用 `dig`（见列表 9-9）。

列表 9-9 用 dig 查询 DNS 服务器

```
$ dig www.google.com
; <<>> DiG 9.5.0-P2 <<>> www.google.com
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 40553
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 7, ADDITIONAL: 7

;; QUESTION SECTION:
;www.google.com.                IN      A

;; ANSWER SECTION:
www.google.com.      497971 IN      CNAME   www.l.google.com.
www.l.google.com.    217     IN      A       66.249.89.99
www.l.google.com.    217     IN      A       66.249.89.104
www.l.google.com.    217     IN      A       66.249.89.147

;; AUTHORITY SECTION:
l.google.com.        27182  IN      NS       e.l.google.com.
l.google.com.        27182  IN      NS       g.l.google.com.
l.google.com.        27182  IN      NS       f.l.google.com.
l.google.com.        27182  IN      NS       a.l.google.com.
l.google.com.        27182  IN      NS       c.l.google.com.
l.google.com.        27182  IN      NS       d.l.google.com.
```



```
l.google.com.      27182  IN      NS      b.l.google.com.

;; ADDITIONAL SECTION:
e.l.google.com.    65980  IN      A        209.85.137.9
f.l.google.com.    65980  IN      A        72.14.235.9
g.l.google.com.    65981  IN      A        74.125.95.9
a.l.google.com.    152381 IN      A        209.85.139.9
b.l.google.com.    152381 IN      A        74.125.45.9
c.l.google.com.    26586  IN      A        64.233.161.9
d.l.google.com.    65980  IN      A        74.125.77.9

;; Query time: 29 msec
;; SERVER: 192.168.1.1#53(192.168.1.1)
;; WHEN: Wed Mar 11 15:39:55 2009
;; MSG SIZE rcvd: 324
```

从列表 9-9 可以看到，dig 以明显的几段输出查询结果。首先是与正在运行的命令相关的一些信息，包括查询是否成功的信息。接下来是查询部分，它显示了实际发送给 DNS 服务器的内容。本例在寻找主机 `www.google.com` 的一个 A 记录。A 记录是把域名映射到地址的记录。我们很快就会详细地介绍记录类型。

回答部分保存对查询的回应。在本例中，它说 `www.google.com` 是 `www.l.google.com` 的一个 CNAME 记录，并且 `www.l.google.com` 分配有 3 个 A 记录。

在权威部分，dig 列出了此次查询的权威名称服务器。在这里可以看到对 `l.google.com` 区域的权威回可以从至少 7 个 DNS 服务器获得，这 7 个服务器的名字从 `a.l.google.com` 到 `g.l.google.com`。dig 在附加部分提供了这 7 个服务器的 IP 地址。

最后，dig 告诉我们此次查询花费了多长时间，查询了哪个服务器，什么时间运行的查询，以及它接收了多少数据。

实际的回应数据显示为 5 栏。这个格式和伯克利 Internet 名称域（Berkeley Internet Name Domain，BIND）内部定义域名的方式一样，它们那里的记录用 5 个字段，分号用来表示注释。这 5 个字段是记录名、距离数据过期的时间（以 `time to live` 或 TTL 为大家所知）、记录类别（对于 Internet 它实际上总是 IN）、记录类型和此记录的数据。

使用 dig，你还可以向任何 DNS 服务器查询具体的记录类型。表 9-1 列出了最常用的记录类型。稍后我们也会设置其中的一些类型。

表 9-1 DNS 记录类型	
类型	用途
SOA	为域定义一个序列号和到期信息
A	把一个主机名映射到一个地址
CNAME	为一个已有的 A 记录添加一个另外的名字
MX	为域指定邮件服务器
NS	为域指定 DNS 服务器
PTR	把一个地址映射到一个主机名

装备了这些知识之后，现在你就可以利用 dig 更高级的功能了。前面只用一个主机名为

参数调用它，但是一个完整的命令通常是这样的：`dig @server name type`。对于第一个例子，完整明确的命令应该是 `dig @192.168.1.1 www.google.com A`。

■注：要使用 `host` 实用程序完成同样类型的查询，请输入 `host -v -t type name server`。

前面通过 `whois` 找出了 `google.com` 域的 DNS 主服务器。为了检验这些 DNS 服务器配置是否合适，你可以向它们查询 `google.com` 域里所有 NS 类型的记录，如列表 9-10 所示。

列表 9-10 向 DNS 服务器查询特定的记录类型

```
$ dig @ns1.google.com google.com NS

; <<>> DiG 9.5.0-P2 <<>> @ns1.google.com google.com NS
; (1 server found)
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 18249
;; flags: qr aa rd; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 4
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;google.com.                IN      NS

;; ANSWER SECTION:
google.com.                 345600 IN     NS     ns1.google.com.
google.com.                 345600 IN     NS     ns3.google.com.
google.com.                 345600 IN     NS     ns4.google.com.
google.com.                 345600 IN     NS     ns2.google.com.
.
;; ADDITIONAL SECTION:
ns1.google.com.             345600 IN     A      216.239.32.10
ns2.google.com.             345600 IN     A      216.239.34.10
ns3.google.com.             345600 IN     A      216.239.36.10
ns4.google.com.             345600 IN     A      216.239.38.10

;; Query time: 178 msec
;; SERVER: 216.239.32.10#53(216.239.32.10)
;; WHEN: Wed Mar 11 17:15:16 2009
;; MSG SIZE rcvd: 164
```

上面的列表显示 DNS 服务器 `ns1.google.com` 的确有 `google.com` 域的 4 个域名服务器的信息，因此它的配置看来是正确的。

3. 区域元数据

较早的时候提到 `dig` 的结果列表中有一栏是 TTL。这个字段定义 DNS 记录的有效期，它允许本地程序缓存 DNS 查询结果一段时间。这样就没有必要为所做的每个连接执行多次 DNS 查找了（记住，先执行一次或多次查找以找到一个权威 DNS 服务器），相当大地加快了网络连接的建立过程。

另一个重要的类型叫作 SOA，它是 Start of Authority 的简写。这个记录包含区域的元信息。例如，它包括一个序列号，服务器可以用它检查区域是否改变，另外它还还为服务器管理

员指定了一个联系人电子邮件。

让我们向其中一个 Google 服务器查询 google.com 域的 SOA 记录（见列表 9-11）。这里去掉了输出的权威部分和附加部分。

列表 9-11 向 DNS 服务器查询 SOA 记录

```
$ dig google.com @ns1.google.com SOA

; <<>> DiG 9.5.0-P2 <<>> google.com @ns1.google.com SOA
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 32611
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 4
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;google.com.                IN      SOA

;; ANSWER SECTION:
google.com.                 86400   IN      SOA      ns1.google.com. dns-admin.google.com.
                2009031200 7200 1800 1209600 300
```

列表 9-11 显示 SOA 记录由 7 个字段组成，它们规定了其他 DNS 服务器如何与这个区域交互。稍后会讲它们，但是目前要提一下列表的最后一项，就是负缓存（negative cache）TTL。它被用来阻止服务器对不存在主机所执行的连续查找，或者用来缓存一个本地回应。本例允许一个远程服务器在初始查询之后 300s 内保持“没有这样的主机”的回应。

9.2.3 运行缓存 DNS

不是所有 ISP 的域名服务器都一样的可靠，有些可能会慢，那么为什么不运行自己的服务器呢？可用的 DNS 服务器软件包有多个，但最常用和最著名的是 Barkeley Internet 域名（BIND）。

■注：BIND 是以它的开发地加利福尼亚大学 Barkeley 分校命名的。

该软件在 Red Hat 上由 bind 软件包提供，基本配置装载在 caching-nameserver 软件包。配置接口由 system-config-bind 提供。它们的安装通过 yum install bind caching-nameserver system-config-bind 完成。在 Ubuntu 上，这些是由 bind9 软件包提供的，你可通过 sudo aptitude install bind9 添加它。DNS 服务器的二进制代码本身叫作 named。

Ubuntu 上所装载的主要配置文件是/etc/bind/named.conf，而在 Red Hat 上使用的是/etc/named.conf。列表 9-12 显示了装载在 Ubuntu 上的基本文件。

列表 9-12 Ubuntu 上的/etc/bind/named.conf 文件前面部分内容

```
// This is the primary configuration file for the BIND DNS server named.
//
```



```
// Please read /usr/share/doc/bind9/README.Debian.gz for information on the
// structure of BIND configuration files in Debian, *BEFORE* you customize
// this configuration file.
//
// If you are just adding zones, please do that in /etc/bind/named.conf.local

include "/etc/bind/named.conf.options";
```

该文件包含对其他文件的引用，而那些文件包含实际的配置设置和本地主机域的信息（可选）。那些配置文件以双斜线（//）为前缀，所有的指令和块以分号（;）结尾。在列表 9-11 中已经包含了 `named.conf` 的一部分。

`include` 指令告诉 `named` 去读指定的文件并处理它所包含的指令，包括嵌套的 `include` 命令。在本例中，`named.conf.options` 文件包含了影响 `named` 工作方式的选项部分。这正是在 Ubuntu 上对配置做改动时所应该编辑的文件（见列表 9-13）。

列表 9-13 Ubuntu 上缺省的 `named` 选项

```
options {
    directory "/var/cache/bind";

    // forwarders {
    // 0.0.0.0;
    // };

    auth-nxdomain no; # conform to RFC1035
    listen-on-v6{ any; };
};
```

`directory` 指令决定 `named` 用来查找文件的位置，如果它写文件的话，也是在这个位置。可以通过指定以/开头的完整系统路径用单个文件覆盖它。

`forwarders` 是 `named` 对上游 DNS 服务器的称呼。如果希望缓存域名服务器只使用 ISP 的域名服务器或一组其他域名服务器，你可以在 `forwarders` 块中列出它们的 IP 地址，每个地址单独一行，并以分号结束。

下一个选项 `auth-nxdomain` 被设置为 `no`。它决定域名服务器如何回应对它认为不存在的域的查找，这意味着本地 DNS 服务器如果不能找到某个域的信息，它就不会声称拥有权威。反过来，这意味着如果这一个服务器没有找到域的信息，客户机可以继续查询其他的 DNS 服务器。

最后，`listen-on-v6` 选项告诉 BIND 监听所有网络接口上所有可用 IPv6 地址的查询。

为了避免先有鸡还是先有蛋的问题，缓存服务器装载了一个内置的根服务器列表。你可以在 Red Hat 上 `/var/named/named.root` 里或者 Ubuntu 上 `/etc/bind/db.root` 里找到它们。你还可以使用 `dig` 向一个根服务器查询“.”区域里所有 NS 类型的记录，以获得当前的根服务器列表。

```
$ dig @a.root-servers.net . NS > db.root.
```

Red Hat 的默认配置文件要通过运行一次 `system-config-bind` 来产生。开启它的方式是选择“系统”→“管理”→“服务器设置”→“域名服务系统”，如图 9-2 所示。

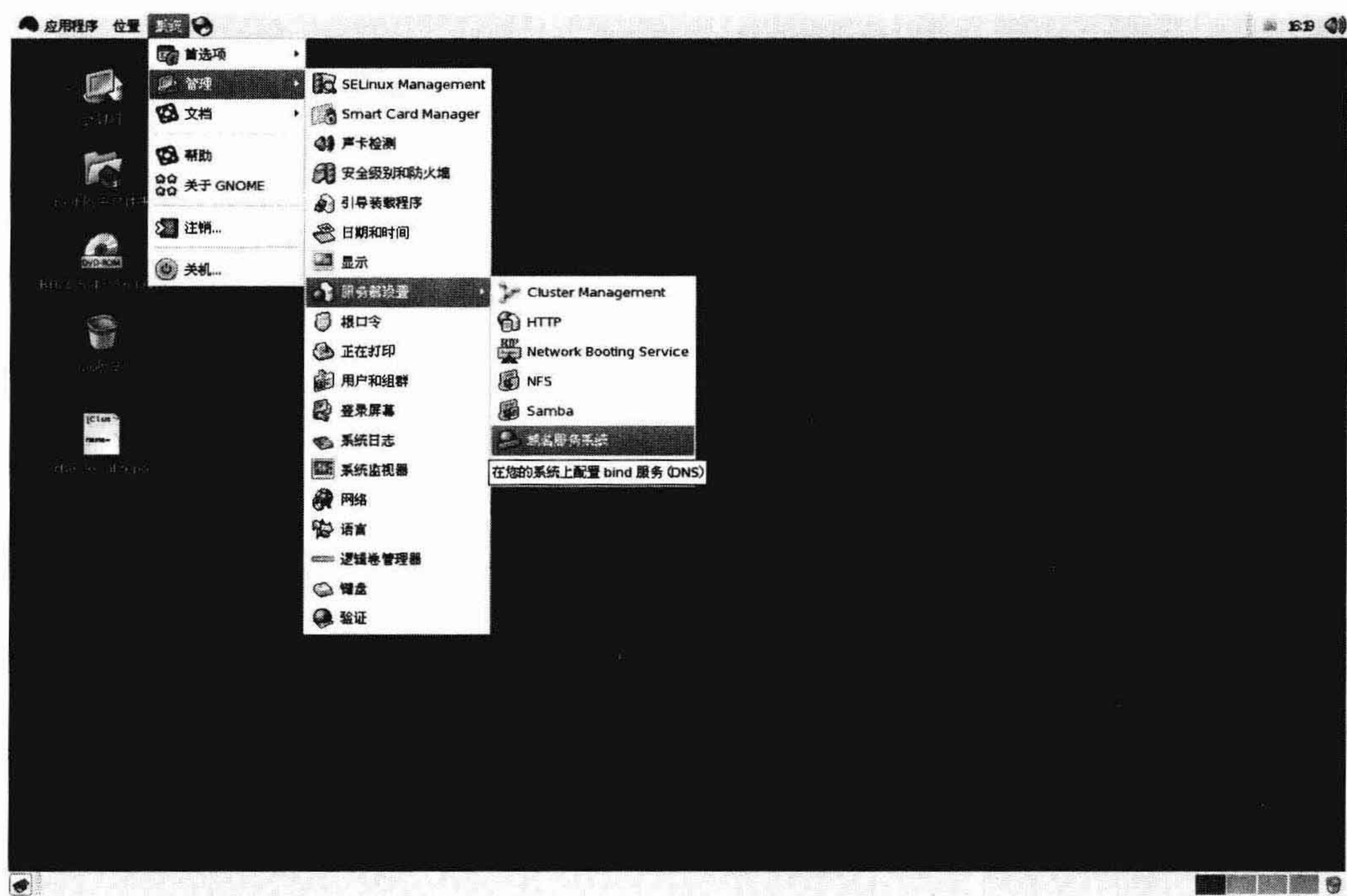


图 9-2 开启 system-config-bind

这个工具需要以 root 用户身份运行，因此在它要求的时候输入 root 密码。

当第一次运行时，它会检测到/etc/named.conf 文件不存在，并在用户单击“确定”的情况下生成一个缺省文件，如图 9-3 所示。然后默认的配置文件被保存到磁盘并加载到 BIND 配置的图形界面。

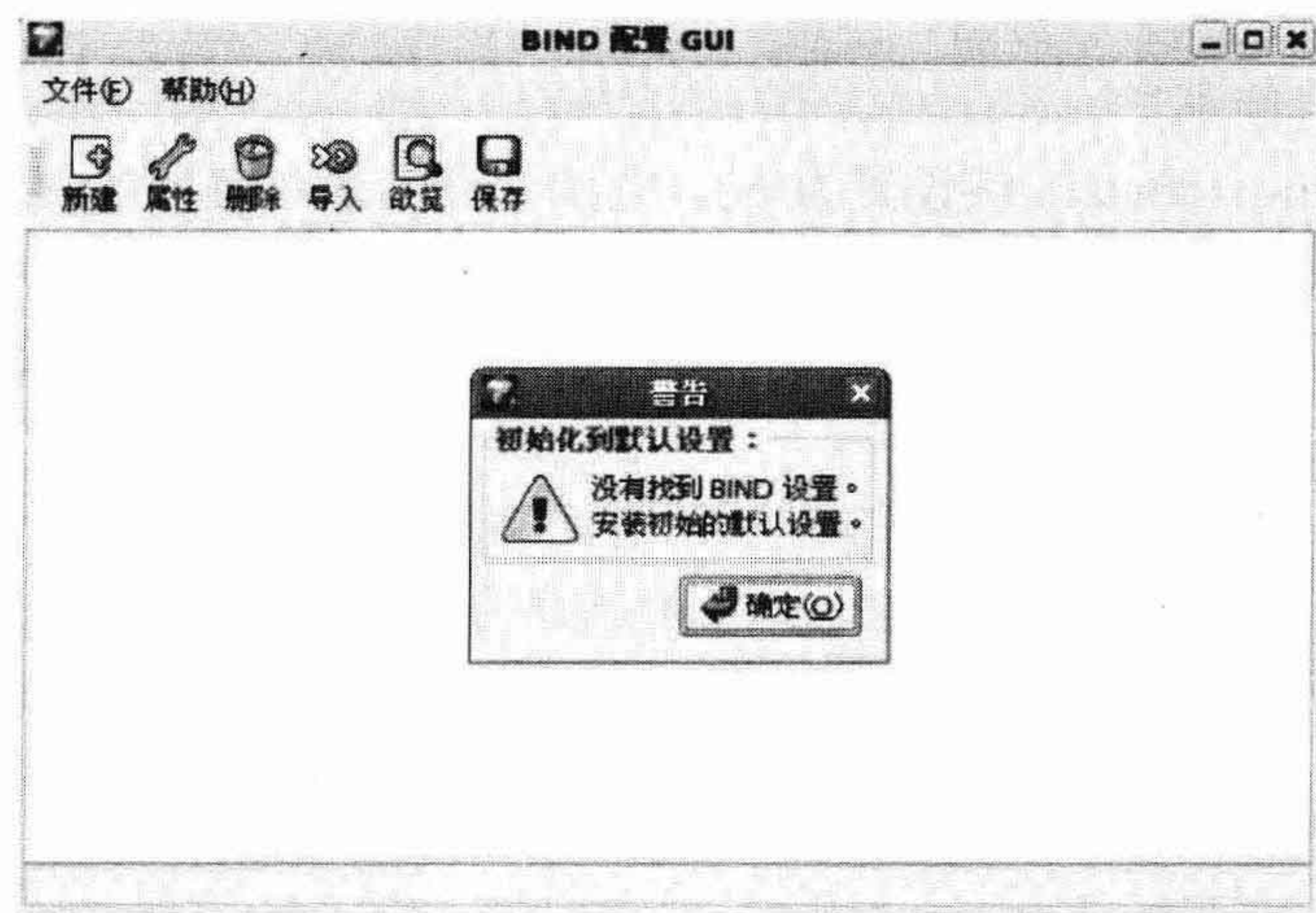


图 9-3 默认配置

任务完成，你可以关掉 BIND 配置 GUI 窗口了。一个基本的/etc/named.conf 文件已经被创建。列表 9-14 中包括了它的重要部分。

列表 9-14 Red Hat 上的/etc/named.conf 文件

```
// Red Hat BIND Configuration Tool
//
// Default initial "Caching Only" name server configuration
//

options {
    directory "/var/named";
    dump-file "/var/named/data/cache_dump.db";
    statistics-file "/var/named/data/named_stats.txt";
    /*
    * If there is a firewall between you and nameservers you want
    * to talk to, you might need to uncomment the query-source
    * directive below. Previous versions of BIND always asked
    * questions using port 53, but BIND 8.1 uses an unprivileged
    * port by default.
    */
    // query-source address * port 53;
};
```

Red Hat 和 Ubuntu 上的文件之间的主要差别在于 `named` 数据的保存位置。`dump-file` 指令允许 `named` 把临时数据写入一个文件，如果数据存在的话。然后它可以在再次启动时重读此数据。`statistics-file` 规定了 `named` 把它接收到的类型和号码查询的相关信息写入的位置。

现在你可以通过 `sudo service named start` 命令开启域名服务器。在 Ubuntu 上，域名服务器在安装时自动开启，但如果它没有运行，你可以通过 `sudo invoke-rc.d bind9 start` 命令开启它。

为了能查询 DNS 服务器，你需确保防火墙没有阻止流量。DNS 默认使用端口 53 上的 UDP 协议，但是如果回应中包含大量数据，它会转到 TCP 协议。根据网络布局向恰当的 Netfilter 链添加合适的规则。

```
$ sudo /sbin/iptables -t filter -A Firewall-eth0-INPUT -p udp --dport 53 -j ACCEPT
$ sudo /sbin/iptables -t filter -A Firewall-eth0-INPUT -p tcp --dport 53 -j ACCEPT
```

■注：你还要确保配置 DNS 主机上的防火墙以允许 DNS 回应。第 6 章讲述了防火墙和 iptables。

现在有了自己的缓存 DNS 服务器，你可以用它查找主机。称它为缓存 DNS 是因为它保存所做的任何查询的回答，从而在下次执行相同的查询时，它可以用缓存的信息立刻做出回应。

为了确保它正常工作，我们直接查询它，如列表 9-15 所示。

列表 9-15 查询本地缓存 DNS 服务器

```
$ host www.google.com localhost
www.google.com is an alias for www.l.google.com.
www.l.google.com has address 74.125.47.99
www.l.google.com has address 74.125.47.147
```



```
www.l.google.com has address 74.125.47.104
www.l.google.com has address 74.125.47.103
```

我们要求运行在 localhost 上的 DNS 服务器查找 `www.google.com` 的地址，它回应了，因此它在起作用！

有了一个正在工作的缓存 DNS，我们就可以用 `nameserver 192.168.0.1` 替换 `/etc/resolv.conf` 文件里的 `nameserver` 输入项来使用我们自己的服务器。我们还可以把这个 DNS 服务器添加到其他任何主机的 `resolv.conf` 文件中。

9.2.4 权威 DNS 服务器

权威 DNS 服务器可替代缓存 DNS 服务器。这意味着这个服务器是一个区域的权威信息源。我们可以使用一个权威服务器为本地网络提供 DNS 解析服务，在本地网络上我们使用 `example.com` 域。为此，定义两个区域：一个提供从域名到地址的映射，一个提供相反的映射，从地址到域名。

为安全起见，不要在提供权威 DNS 服务器的同时使用缓存 DNS 服务器。

1. 调整区域

区域在区域文件中定义，与较早前所提的根区域文件很类似。区域文件总是包含一个报头，也就是著名的 SOA 记录。在这个报头后面你可以选择跟随定义服务和主机的 DNS 记录。列表 9-16 里包含了一个简单的区域文件报头。

列表 9-16 example.com 域的区域文件报头

```
$ORIGIN example.com.
$TTL 86400
@ IN SOA example.com. root.example.com. (
    2009013101 ; Serial
    604800 ; Refresh
    86400 ; Retry
    2419200 ; Expire
    3600 ) ; Negative Cache TTL
```

这个报头定义了一些与我们的区域相关的元信息，这个信息由缓存 DNS 服务器以及可能定义过的从服务器使用。从服务器 (slave servers) 是权威 DNS 服务器，它们自动从一个主 DNS 服务器检索它们的区域信息。你可以使用它们提供冗余 DNS 服务，就像 ISP 所做的那样。

表 9-2 中列出了区域报头的字段及其用途。在我们的例子中，所列出的时间都是以秒为单位，但是还可以使用 `1d` 代替 `86400` 表示 1 天，或者用 `4w` 代替 `2419200` 表示 4 周。

表 9-2 区域报头的字段

字段	用途
\$ORIGIN	定义区域的开头
\$TTL	生存期，是此区域中没有为自己设置到期时间的那些记录的默认到期时间

续表

字段	用途
SOA	Start of Authority，它包含区域元数据的 7 个记录
Master	此域的第一 DNS 权威服务器
Contact	此域联系人的电子邮件地址，其中的 at 符号（@）用句点代替
Serial	定义此区域文件的版本，由从域名服务器使用
Refresh	定义从服务器更新此区域副本的频率
Retry	定义两次刷新从服务器的尝试之间的时间间隔
Expire	定义允许从服务器使用此区域文件某个版本的时间
Negative Cache TTL	定义不成功的查找结果可以缓存的时间

同样值得注意的是，该区域正在使用的是基于日期的序列号。格式是 YYYYMMDDNN，即当前的年月日跟着一个两位数。通过它可轻易看到该区域上次修改的时间，而且还容许每天修改 99 次。从列表 9-11 中可以看到 Google 使用了这个格式。作为替代，你可以使用一个简单的递增数字作为序列号，这是 Red Hat 上 system-config-bind 实用程序所使用的方法。SOA 前面的 at 符号（@）表示当前区域的名字。在这个地方你还可以键入 example.com。

2. 正向查找区域

与 Red Hat 相反，在 Ubuntu 上创建区域有点费力。你要通过一个文本编辑器创建区域文件，并把它们的定义添加到/etc/named.conf.local 文件，而在 Red Hat 上可以使用所提供的管理程序。先看看手动方法。

我们打算把正向查找区域保存在一个叫 example.com 的文件里。因为只有 root 用户可以写入区域文件的目录，因此用 sudo 开启编辑器。

```
$ sudo vim /var/cache/bind/example.com.db
```

现在，我们直接把列表 9-17 里的区域报头复制粘贴到这个文件里并保存它。报头完成后，我们可以开始往文件里添加实际的主机和服务记录。

你需要在区域里提供两个基本的服务记录类型。一个是 NS 记录，定义哪一个主机担当此域的 DNS 服务器角色，另外一个 MX 记录，定义此域的邮件服务器。这两类记录都以空白字段开头，因为它们没有定义主机名。

列表 9-17 服务记录

```
NS ns.example.com.
MX 10 mail.example.com.
```

MX 记录的数据包括一个优先级数字以及远程服务器应该尝试投递的邮件目的地主机名。配置恰当的远程邮件服务器将通过 MX 记录列表从最低优先级数字开始工作，并试图发送电子邮件。注意，要指定一个输入项末尾带句点的完全合格域名（fully qualified domain name, FQDN；它是主机名加全域名）。如果把它漏掉了，DNS 服务器会假定只定义了主机名，它会自动把\$ORIGIN 追加到这些记录的结尾。

这些定义中使用了 ns 和 mail 主机名，但是还没有在区域文件里定义这些主机，那么接

下来让我们把它完成（见列表 9-18）。主机到地址的记录被称为 A 记录。我们还要为当前的主机名添加一个 A 记录。

列表 9-18 为域添加 A 记录

```
@ IN A 192.168.0.1
ns IN A 192.168.0.254
mail IN A 192.168.0.1
au-mel-ubuntu-1 IN A 192.168.0.1
```

这里没有在这些记录的主机一栏里指定一个 FQDN，因此 DNS 服务器会把它们看作与追加了 \$ORIGIN（example.com）一样，这正是我们想要的结果。@符号也被替换为它的来源，因此用户只到此域也可以访问一个主机。你可以看到这两个域名现在解析为相同的地址。与 IP 地址相关联的 A 记录想要多少就可以有多少。正向区域里另外一个类型或记录被称为 CNAME，也就是大家所知的别名。你可能记得较早前查找 www.google.com 时发现别名的事情。

如果想给单个主机关联多个别名，并希望在无需修改 A 记录的长列表的情况下仍能够修改那个主机的地址，你就可以使用 CNAME。例如，主机 au-mel-ubuntu-1 要提供网站和 SQL 服务，同样邮件服务器提供对 POP 和 IMAP 的访问。我们可以创建一些 CNAME 输入项以提供全指向 mail A 输入项的别名（见列表 9-19）。如果将来把邮件服务转移到另一个主机，我们只需修改 A 记录，所有 CNAME 输入项会自动指向新地址。

列表 9-19 添加一些 CNAME 输入项

```
gateway IN CNAME ns.example.com.
headoffice IN CNAME au-mel-ubuntu-1.example.com.
smtp IN CNAME mail.example.com.
pop IN CNAME mail.example.com.
imap IN CNAME mail.example.com.
www IN CNAME au-mel-ubuntu-1.example.com.
sql IN CNAME au-mel-ubuntu-1.example.com.
```

我们还创建了叫作 gateway 和 headoffice 的 CNAME，在第 14 章设置 VPN 时会用到它们。这就是目前所需的全部。保存此文件，并创建一个与之相伴的逆向区域文件以提供地址到域名的映射。

3. 逆向查找区域

为了设置一个逆向区域，你要首先找出它叫什么。与正向区域不同，它没有域名，但有一个唯一的地址范围。为了提供对地址的查找，你要使用一个名为 in-addr.arpa 的域。这本质上是逆向映射的根区域。

与正向区域一样，你要预先考虑此区域网络地址的组成成分，最重要的部分在右边。对于我们的 192.168.0.x 网络，逆向区域名就是 0.168.192.in-addr.arpa.。

■注：in-addr.arpa 区域的前缀总是由多达 4 个数字组成的 IP 地址中的 3 个数字组成。少于 255 个地址的子网没有标准的逆向区域方法。

我们再次启动编辑器（以 root 用户身份）创建一个新区域文件。

```
$ sudo vim /var/cache/bind/192.168.0.db
```


该文件的报头要与正向区域有所不同，因为区域名不同。我们添加列表 9-20 中所示的内容。

列表 9-20 逆向区域的报头

```
$ORIGIN 0.168.192.in-addr.arpa.
$TTL 86400
@ IN SOA ns.example.com. root.example.com. (
    2009013101 ; Serial
    604800 ; Refresh
    86400 ; Retry
    2419200 ; Expire
    3600 ) ; Negative Cache TTL
```

创建完报头，我们现在就可以开始添加把地址映射为域名的 PTR 记录了。让我们为堡垒主机添加一个，为 192.168.0.254 上的主机添加一个，并添加邮件 A 记录，如列表 9-21 所示。

列表 9-21 为主机添加 PTR 记录

```
1 PTR mail.example.com
1 PTR au-mel-ubuntu-1.example.com.
254 PTR ns.example.com.
```

我们保存这个逆向区域文件并退出编辑器。现在剩下要做的事就是把这两个区域的区域定义添加到/etc/bind/named.conf.local 文件里。我们打开这个文件并添加这些定义，如列表 9-22 所示。每个区域指令块包含一个对定义该区域的文件的引用。服务器默认期望这些文件都在主配置文件指定的目录里。既然是提供这些区域的权威 DNS 服务器，我们就需要把这个区域类型设置为主类型。

列表 9-22 添加区域定义

```
zone "example.com" {
    type master;
    file "example.com.db";
};

zone "0.168.192.in-addr.arpa" {
    type master;
    file "192.168.0.db";
};
```

然后我们保存该文件并退出编辑器。我们需要通过 invoke-rc.d 或 service 命令重启服务器或者使用 rndc 工具告诉服务器重新加载它的配置。后者更快并且不中断服务，因此我们使用它来进行。

```
$ sudo rndc reload
```

域名服务器应该知道我们的新区域，我们可以查询它来检查一下。我们从查找 ns.example.com 的地址开始，如列表 9-23 所示。

列表 9-23 测试正向域名解析

```
$ host ns.example.com localhost
Using domain server:
Name: localhost
Address: 127.0.0.1#53
Aliases:
ns.example.com has address 192.168.0.254
```


工作得很好。我们也通过查找与 192.168.0.1 地址相关联的域名检查一下逆向区域，如列表 9-24 所示。

列表 9-24 测试逆向域名解析

```
$ host 192.168.0.1 localhost
Using domain server:
Name: localhost
Address: 127.0.0.1#53
Aliases:

1.0.168.192.in-addr.arpa domain name pointer au-mel-ubuntu-1.example.com.
1.0.168.192.in-addr.arpa domain name pointer
mail.example.com.0.168.192.in-addr.arpa.
```

不是十分正确！域名服务器把逆向区域名追加到 mail.example.com 主机上了。不过我们知道通常是什么原因导致了这个结果，因此如果去检查逆向区域文件，我们就可以看到的确忘掉了 mail.example.com 末尾的句点。

我们马上加上句点并把区域的序列号加 1。完成后，我们再发出 sudo rndc reload 命令。如果再测试逆向解析结果，我们就会看到问题已经解决了。

4. 使用 Red Hat 上的 system-config-bind

在 Red Hat 上添加和修改区域的操作有点不太一样，因为我们可以使用 GUI 工具来做。让我们再次打开它，然后添加 example.com 和 0.168.192.in-addr.arpa 区域的正向和逆向输入项。然后选择“新建”→“网络区域”，如图 9-4 所示。

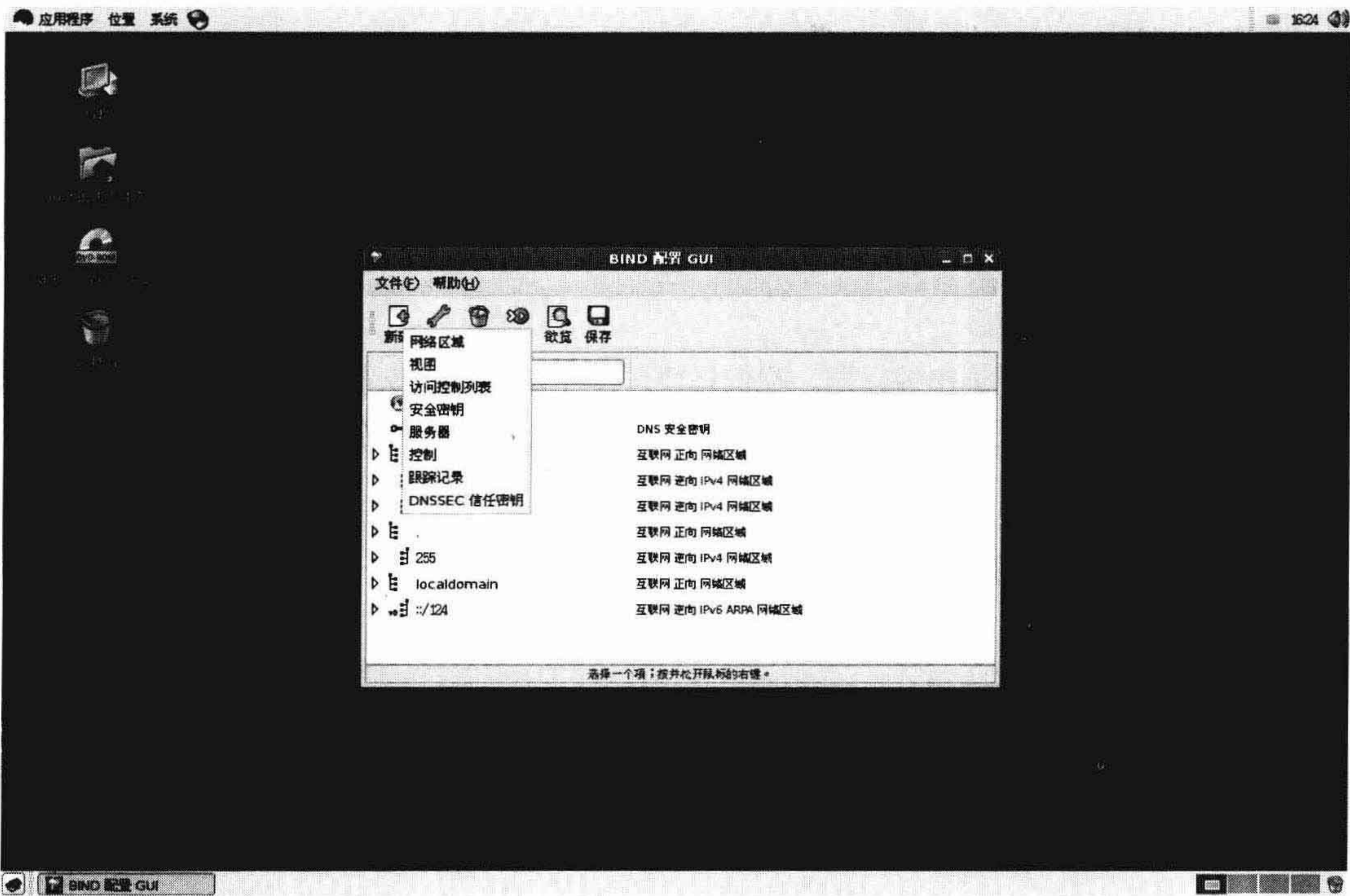


图 9-4 添加新区域

接下来要指定类别、来源和类型。对默认类别和来源类型单击“确定”，如图 9-5 所示，因为我们正在创建一个 Internet 正向区域。

当接受来源类型时，需要输入此区域的来源——它的名字。本例中输入了 example.com，如图 9-6 所示。



图 9-5 指定区域类型



图 9-6 输入区域的来源

区域类型是正确的，因此单击“确定”按钮。然后呈现的是此区域的 SOA 属性对话框，如图 9-7 所示。

较早前说过，它设置区域的元数据和一些从服务器的选项。我们把权威名称服务器设置为 ns.example.com，负责人设置为那个主机的 root 用户。默认的到期时间都正确，因此无需修改。最后是区域文件路径，就是该区域所存储的位置。它在 Red Hat 上是 /var/named，如较早前看到的那样。

既然区域已经创建完，那么我们可以开始给它添加记录了。NS 记录在创建区域时自动完成了，因此接下来添加 MX 记录。单击“example.com. zone”并选择“添加”→“MX 邮件交换”，如图 9-8 所示。

在 MX 记录对话框中，把优先级设置为 10，并输入邮件服务器的名字，如图 9-9 所示。

剩下要做的就是为刚定义的主机创建 A 记录。再次右击区域名，然后选择“添加”→“A IPv4 地址”，如图 9-10 所示。

在接下来的对话框中，添加 FQDN 和希望与此域名相关联的地址，如图 9-11 所示。注意“创建逆向映射记录”复选框。只要保证选定了它，system-config-bind 工具就会自动创建并组装相关联的逆向映射区域。

单击“确定”把这个 A 记录添加到区域中。如果现在看看主窗口，我们可以看到已经添加了一个 192.168.0 区域。当单击箭头打开该区域时，我们可以看到已经创建了一个把 1 映射到 ns.example.com 的 PTR 记录，如图 9-12 所示。

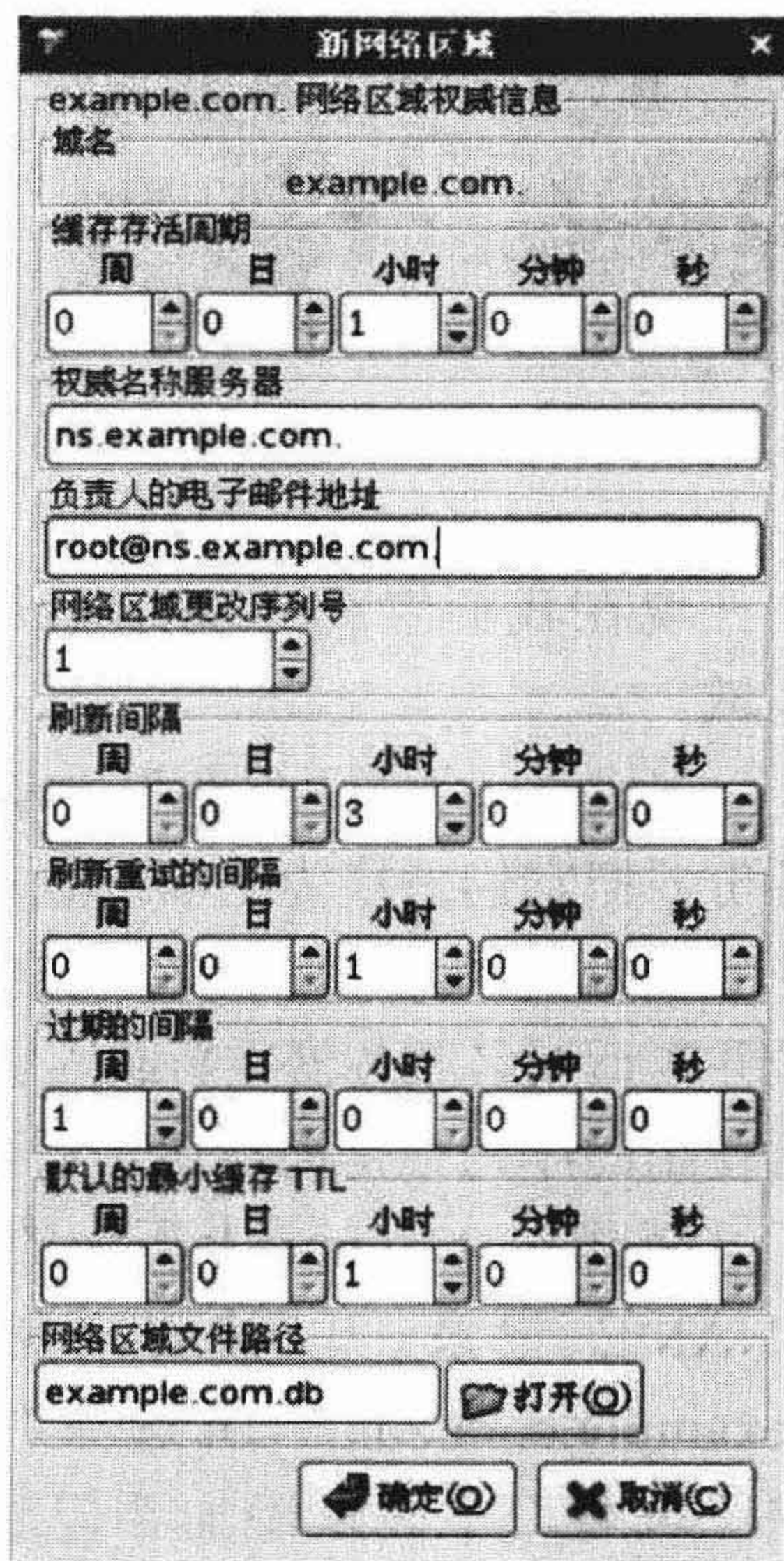


图 9-7 区域的权威信息

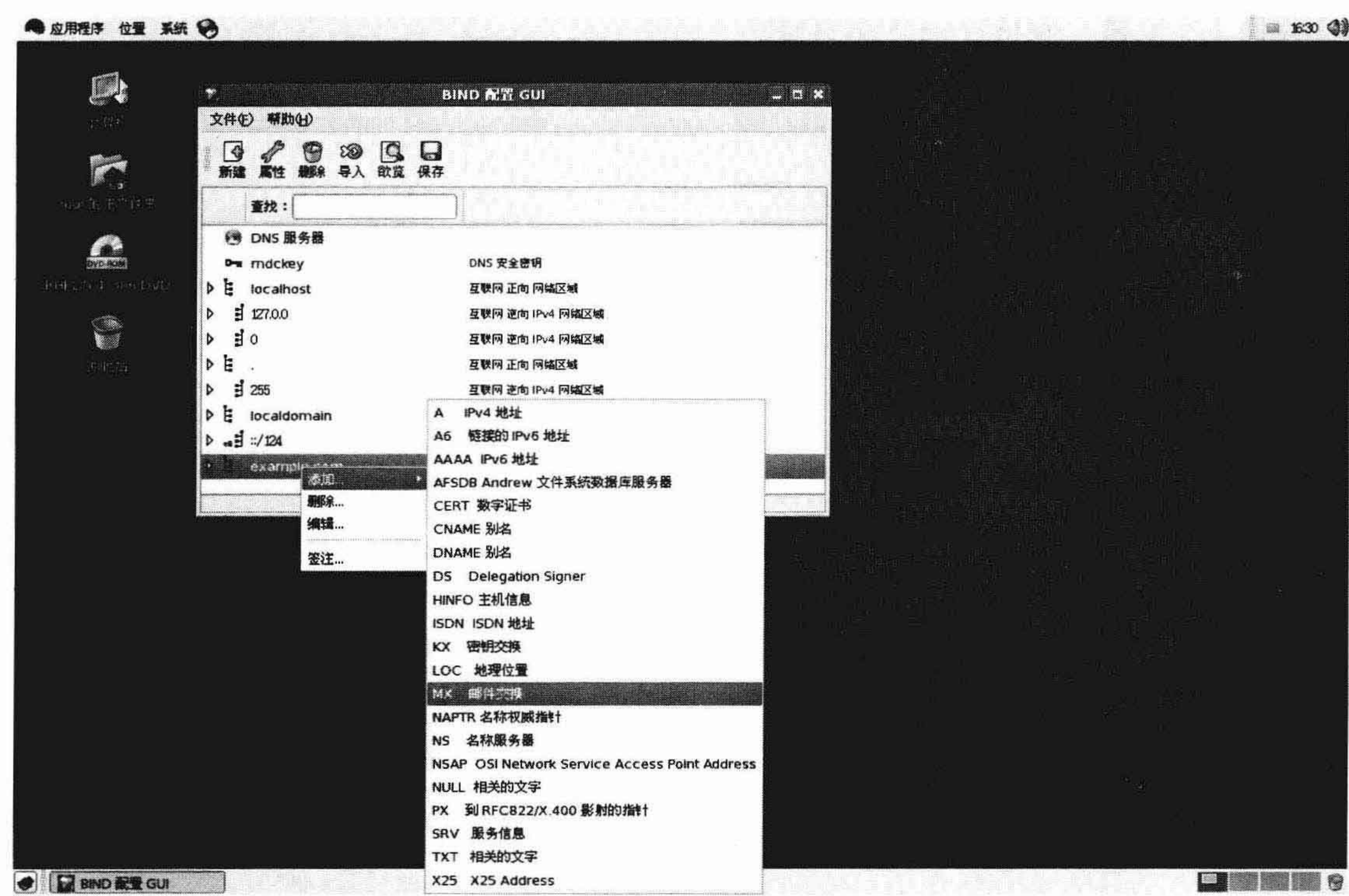


图 9-8 给此区域添加 MX 记录

现在我们继续为 mail.example.com 和 au-mel-rhel-1.example.com 添加 A 记录到正向区域。

完成后，单击“保存”。system-config-bind 工具会问是否希望创建当前配置和区域文件的备份并把它们替换为更新的版本，如图 9-13 所示。

在单击“是”时，该工具会自动增加改动了的区域序列号并告诉域名服务器重新载入它的区域文件。值得注意的是，它使用的序列号是简单递增的数字，而不是基于日期。

就像在 Ubuntu 上所做的那样，现在让我们添加几个 CNAME 记录，以使用户更容易访问服务。高亮显示“example.com.zone”并选择“新建”→“CNAME 别名”。在“域名”字段里添加希望创建的别名，在“标准的名”字段添加该主机的 FQDN，如图 9-14 所示。

我们还为 pop 和 imap 添加了 CNAME 记录，两者都指向 mail.example.com.，然后添加了 www 和 sql，它们指向 au-mel-rhel-1.example.com。最后我们添加了一个指向 ns.example.com 的叫作 gateway 的 CNAME 和一个指向 au-mel-rhel-1.example.com 的叫作 headoffice 的 CNAME。它们会在第 14 章设置 VPN 时用到。

完成后，单击“保存”更新区域文件并重新载入服务器。



图 9-9 设置 MX 记录的资料

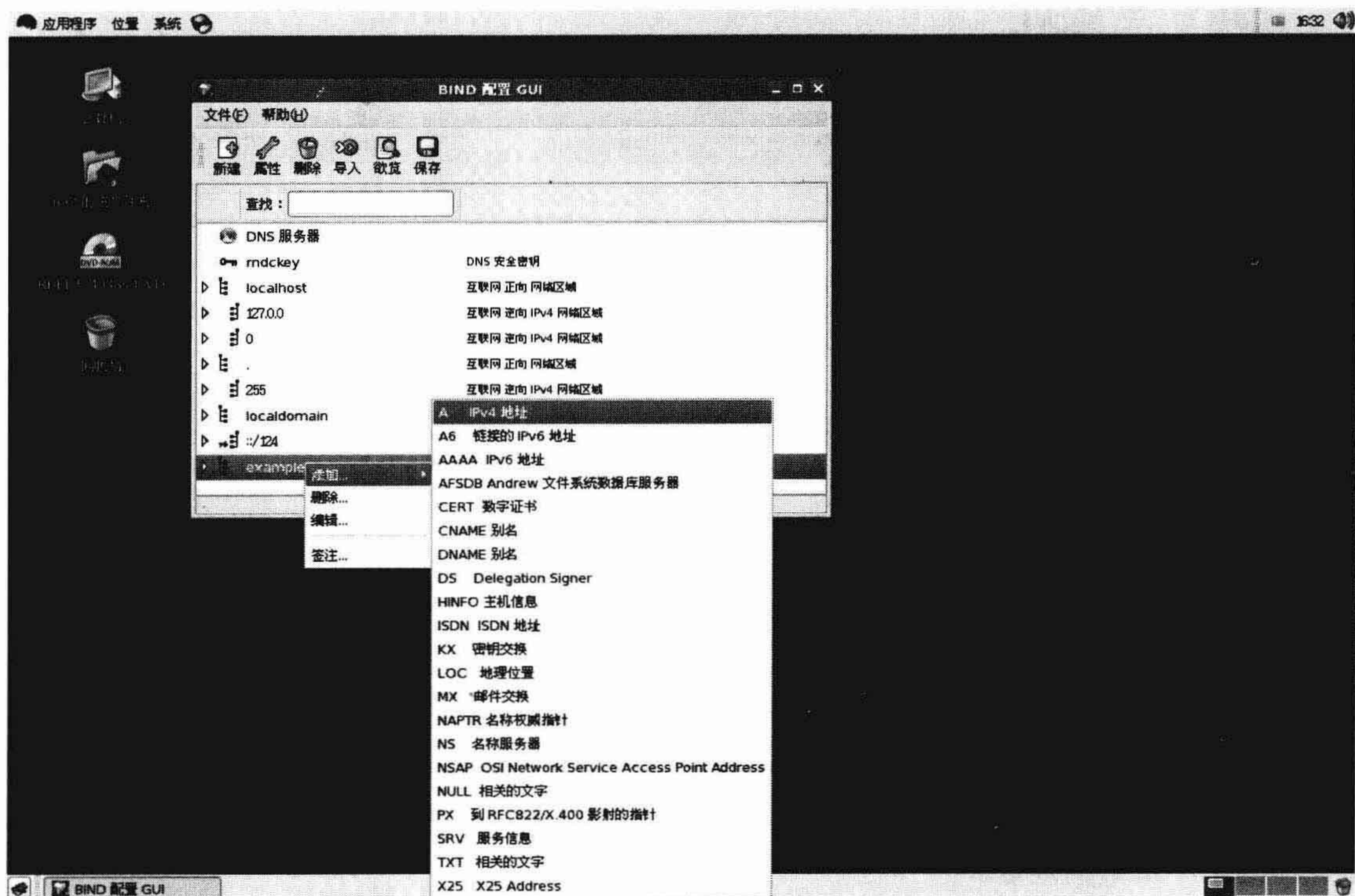


图 9-10 添加 A 记录



图 9-11 新的 A 记录资料

图 9-12 自动创建反向区域

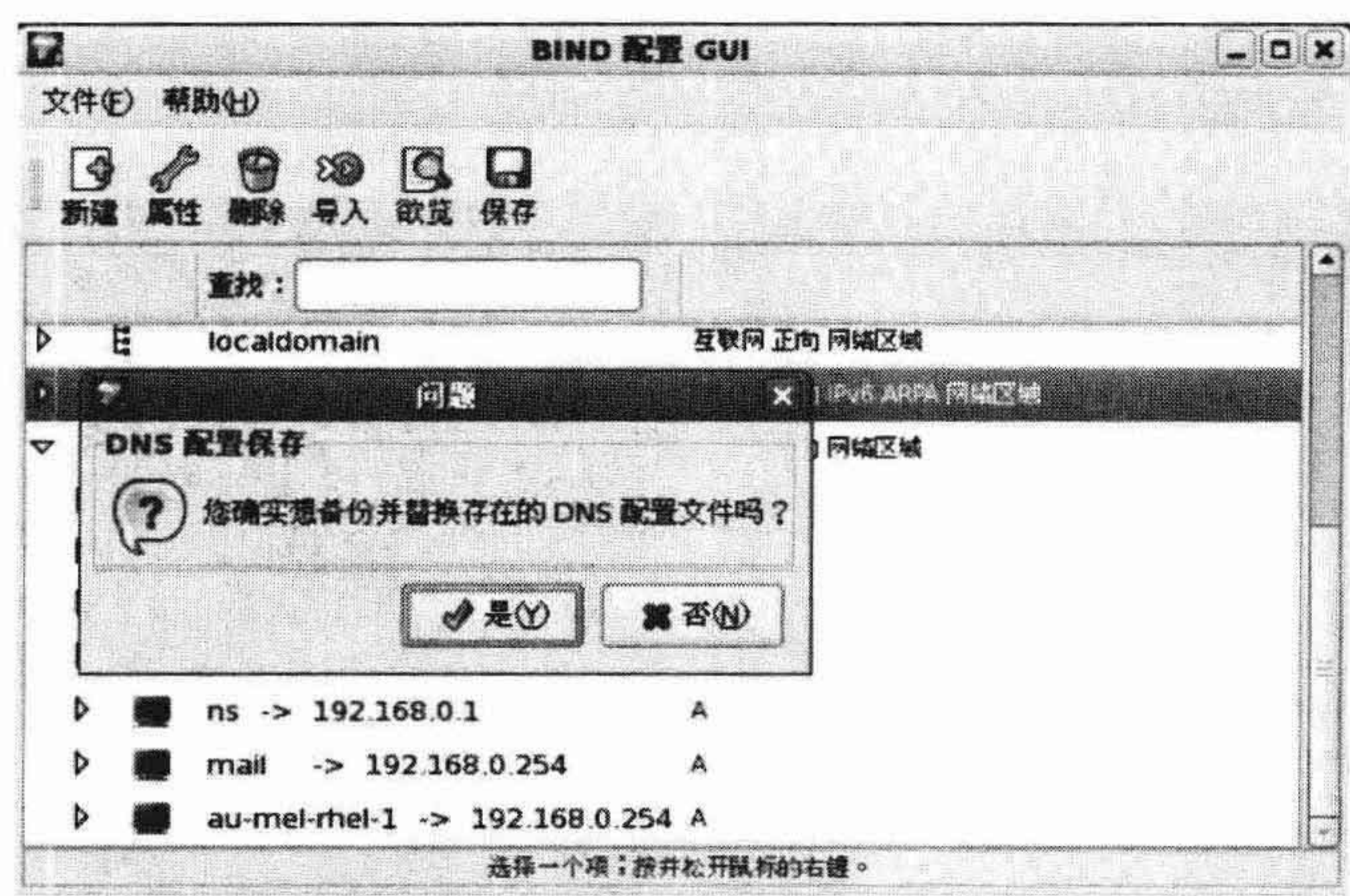


图 9-13 保存更新的配置文件

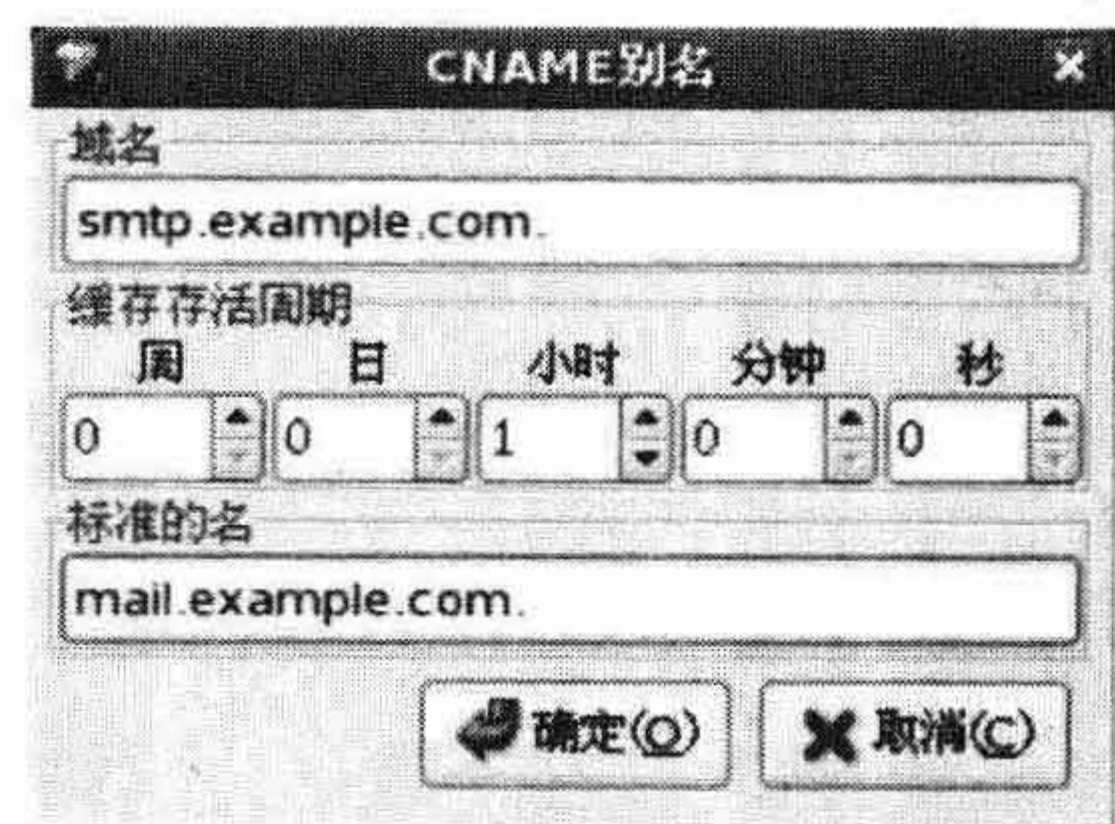


图 9-14 添加 CNAME 记录

5. 安全考虑

现在我们正在把堡垒主机上的 DNS 服务器当作一个权威和缓存 DNS 服务器运行。尽管软件可以把它处理得很好，但还是有一些安全问题要考虑。主要的一个考虑是以 DNS 缓存中毒（cache poisoning）著称的攻击。在这种攻击中，攻击者通过让用户单击一个恶意网站的链接或者打开一个内嵌链接的电子邮件，使它执行一个 DNS 查询，通过这种方式使缓存 DNS 服务器给出不正确的地址。

■注：在 http://en.wikipedia.org/wiki/DNS_cache_poisoning 上可以了解更多有关 DNS 缓存中毒的信息。

通过把自己的权威域放在不查询上游服务器的隔离 DNS 服务器上，我们可以防止这种攻击对该域的影响。这并没有解决问题，但它把对基础架构的影响最小化了。

6. 添加一个从服务器

为了提供可靠的 DNS 服务，实际上所有的域登记机构都会要求为一个域输入至少两个 DNS 服务器。当然维护所有区域文件的多个副本是可能的，但可以利用 BIND 里的主/从功能使这个过程自动化。

让我们从 Red Hat 开始，为 example.com 域添加一个从服务器。

Red Hat

首先，你要让主服务器允许从服务器检索域的区域文件。我们在主服务器上打开 BIND 配置程序并高亮显示“example.com”区域，然后单击“属性”。

在区域选项画面上，我们到右边“所有选项”的列表中找到“allow-transfer”选项，然后单击“+”图标。现在我们高亮显示左边选项列表里的“allow-transfer”选项，并添加从服务器的 IP 地址，如图 9-15 所示。

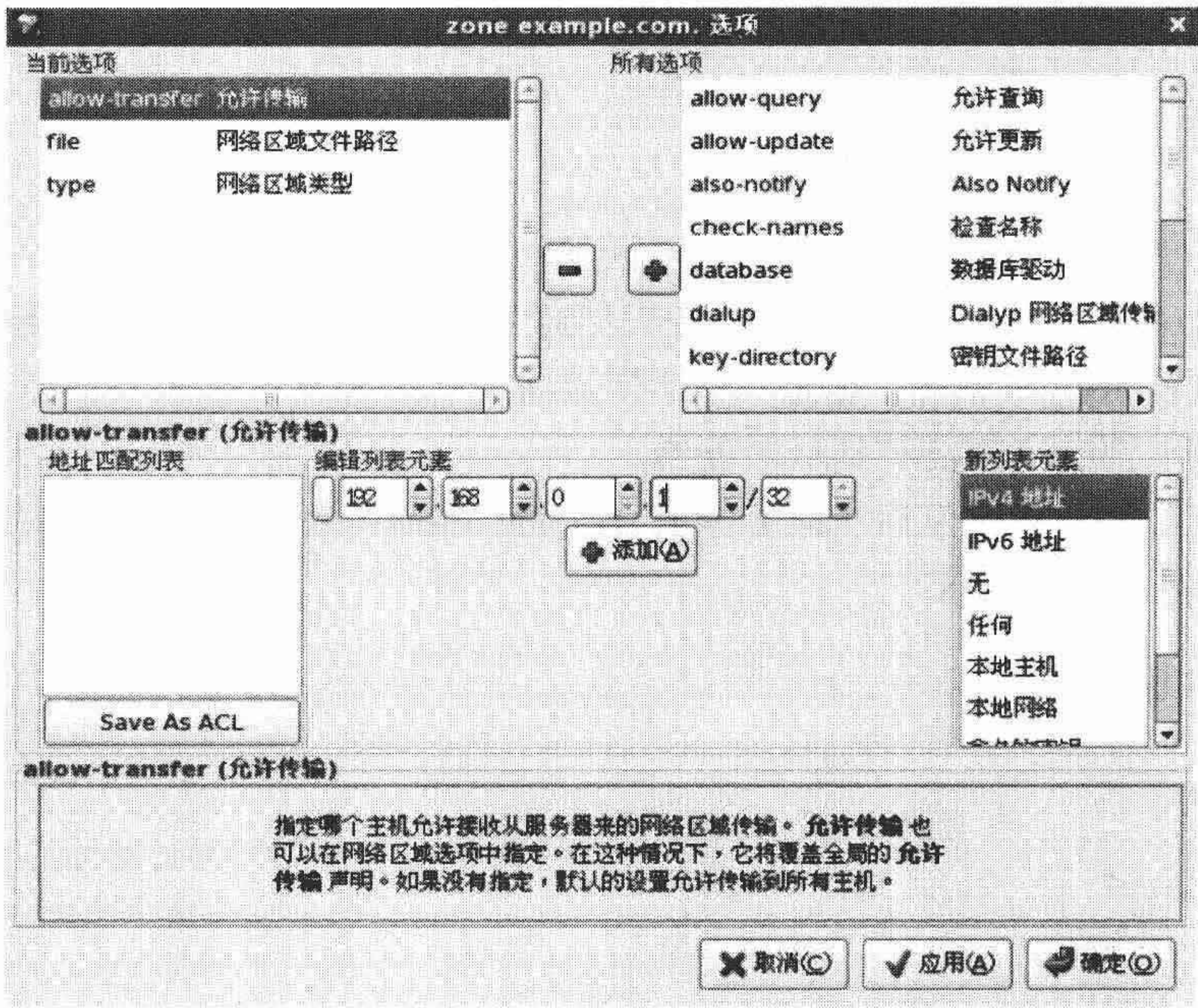


图 9-15 允许从服务器传送区域文件

单击“添加”把地址添加到匹配列表中。如果想添加更多从服务器，我们对每一个从服务器都重复一遍这个过程。接下来，单击“确定”接受这个改动并返回主画面。保存配置以应用改动并重新载入 DNS 服务器。

接下来要配置从服务器。我们在要成为从服务器的主机上打开 BIND 配置工具，选择“新建”→“域”。接受 IN 类别并输入“example.com.”作为区域的来源。对于区域类型，我们选择“slave”，如图 9-16 所示。

单击“确定”接受并进入下一个画面。在这里我们要添加主 DNS 服务器的 IP 地址。高亮显示“IPV4 地址”并输入地址，如图 9-17 所示。



图 9-16 添加从服务器区域

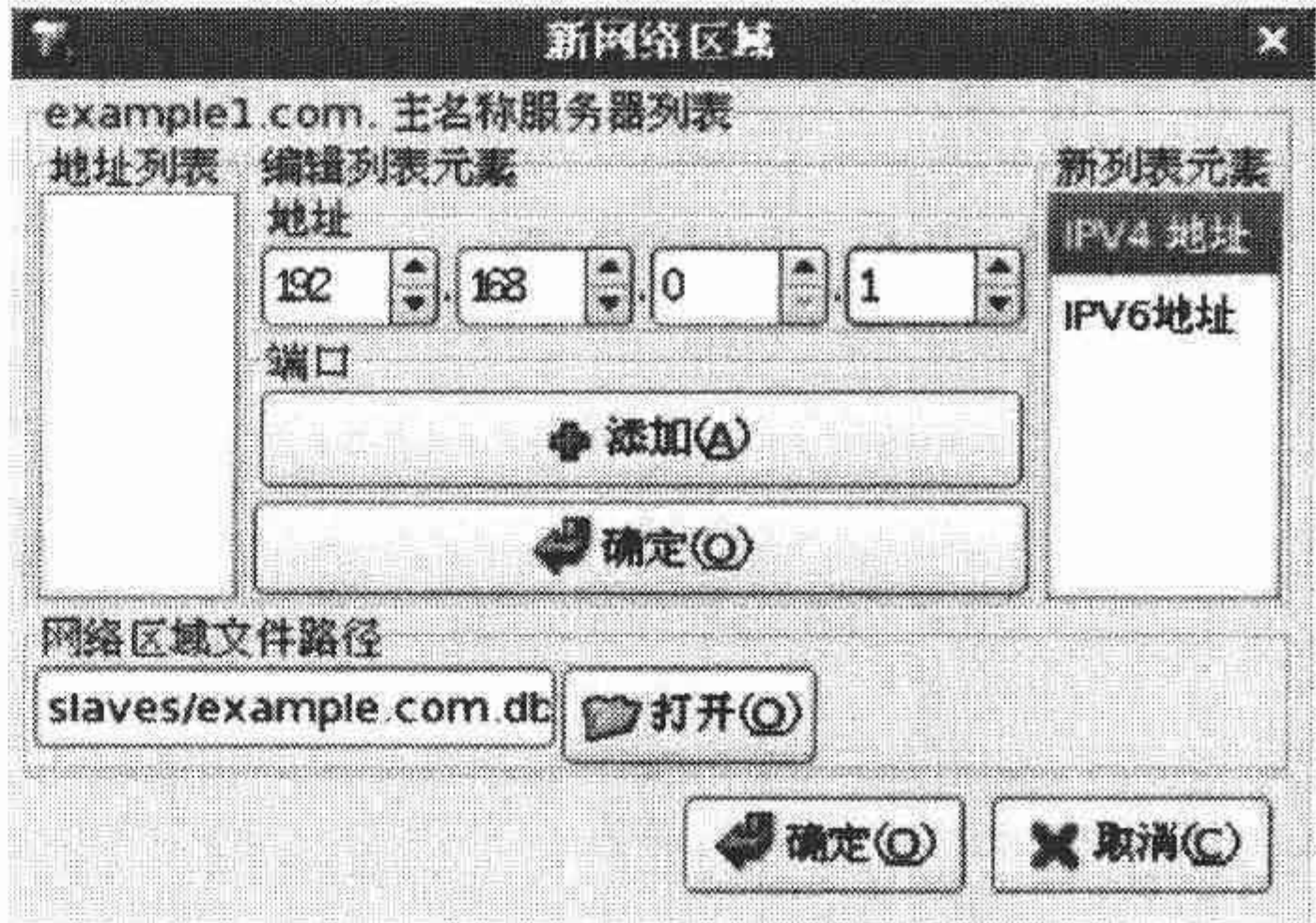


图 9-17 添加主服务器

完成后，单击对话框中间的“确定”按钮把地址添加到左边的列表。然后单击“确定”接受此配置。最后，保存配置并退出程序。

BIND 会立即重启，并同时检索来自主服务器的区域文件。你可以通过/var/log/messages 文件检验这一点。

Ubuntu

如果想在 Ubuntu 上设置主服务器以使它允许来自从设备的传送，我们就要修改/etc/bind/named.conf.local 里的区域定义。我们要确保主服务器在一个区域更新时联系从设备，为此，可以添加 notify yes 指令。这意味着不需要等待从服务器达到区域期满，因为主服务器上的任何区域改动都会立刻复制到从服务器。

接下来我们添加一条要包括从服务器 IP 地址的 allow-transfer 指令。列表 9-25 中包含了“example.com”区域的新定义。

列表 9-25 添加区域定义

```
zone "example.com" {
    type master;
    notify yes;
    allow-transfer{
        192.168.0.2;
    };
    file "example.com.db";
};
```

添加完所有从服务器的地址后，保存文件，然后通过 sudo rndc reload 告诉 BIND 重新载入它的配置。

提示：为了测试主服务器的配置，你可以在从服务器上使用 dig 命令模拟一个区域传送。使用 AXFR 查询类型：dig example.com @192.168.0.1 AXFR。

下一步是告诉从服务器到哪儿找主服务器。我们打开从服务器上的/etc/bind/named.conf.local 文件并为“example.com”域添加一个区域定义。我们设置区域类型为 slave。为了确保服务器可以检索区域数据，我们要在 masters 配置块中指定主服务器的地址。列表 9-26 中包含了我们网络的配置。

列表 9-26 从服务器的区域配置

```
zone "example.com" {
    type slave;
    masters {
        192.168.0.1;
    };
    file "example.com.db";
};
```

完成之后，我们保存配置文件并通过 sudo rndc reload 命令告诉从服务器重新载入它。你可以检查/var/log/syslog 文件来验证区域正在被传送，或者使用 host 或 dig 命令查询从服务器以确保区域数据存在。

9.2.5 动态 DNS

如果 ISP 在你每次连到 Internet 时都给主机分配一个新的随机地址，那么运行自己的权威 DNS 服务器就没有多大意义。服务器地址会一直变动，我们需要一直修改 WHOIS 数据库里的权威信息。

一个替代方案是动态 DNS，Internet 上就有很多种动态 DNS 提供者。在这种方案中，动态 DNS 提供者就是 DNS 服务器的宿主。一个小客户应用程序运行在你系统里的一台主机上。无论你的 IP 地址何时改变，它都会远程更新 DNS 服务器上的主机记录。当然，即使外部 IP 地址不会变，你也可以使用这样一个服务。

存在各种各样的 DNS 提供者，在 http://www.dmoz.org/Computers/Internet/Protocols/DNS/DNS_Providers/Dynamic_DN 上有一个不完全的列表。如果选择用这种方式外包 DNS 功能，那么所选择的 DNS 提供者会提供 DNS 服务器的资料，以便输入到你的登记机构的登记表格中。

如果需要动态更新，那么所选择的提供者应该提供能在 Linux 下工作的客户程序。这些程序中有几个是 Ubuntu 里的软件包，例如 `ddclient` 或者 `ez-ipupdate`。

- `ddclient`: <http://sourceforge.net/projects/ddclient/> 和 <http://dag.wieers.com/rpm/packages/ddclient/>。
- `ez-ipupdate (source)`: <http://www.ez-ipupdate.com/>。

对于 Red Hat，我们需要下载其中一个工具的压缩包（tarball）并手动安装它，或者找一个第三方创建的 RPM 包。

9.3 动态主机配置协议

既然对主机进行了分类命名，那么我们也许应该给某些主机（如工作站或膝上电脑）自动分配网络地址。用于此目的的服务是动态主机配置协议（Dynamic Host Configuration Protocol, DHCP）。它由一个服务器和一个客户端组成，服务器定义哪些地址可以分配给哪些客户端，而客户端向服务器请求地址并利用回应配置本地网络接口。

这对于那些希望随意添加到网络的机器非常好，你不必关心它们分配了什么地址。不过，对于服务器，即使使用 DHCP，一般还是希望静态分配。如果服务器的地址不可预知地改变，我们将不能使用它所提供的服务。

幸运的是，DHCP 服务器允许把可用的网络地址范围分成一些地址池。然后每一个地址池可以被配置分配给已知的主机或者未知的主机。这样，就有可能让来访的膝上电脑分配到一个特定范围内的随机空闲地址。

9.3.1 安装与配置

DHCP 服务器在 Ubuntu 上由 `dhcp3-server` 软件包提供，在 Red Hat 上由 `dhcp` 软件包提供。

Ubuntu 上安装了一个配置文件样例 `/etc/dhcp3/dhcpd.conf`。Red Hat 使用默认空白的 `/etc/dhcpd.conf` 文件，不过它装载了 `/usr/share/doc/dhcp-3.0.5/dhcpd.conf.sample` 文件。

这个配置文件由一组全局指令和随后的 1 个或更多子网定义组成。注释以 `#` 为前缀。列表 9-27 中包含了来自 Ubuntu 文件的全局指令。

列表 9-27 dhcpd.conf 全局设置

```
ddns-update-style none;
default-lease-time 600;
max-lease-time 7200;
log-facility local7;
```

第一个指令规定 DHCP 服务器不要为它所分发的地址做 DNS 更新。稍后你就会看到如何修改这一点。`default-lease-time` 指令规定一个正在连接的客户端没有指定时间的情况下的 DHCP 租约有效时间。如果客户端指定了一个时间，这个时间不能超过 `max-lease-time`。这两个时间都是以秒为单位。最后，`log-facility` 指定系统记录器应该如何处理由 DHCP 服务器生成的记录输入项。第 18 章会介绍系统日志的配置方法。

让我们稍微修改一下这个配置，以使它适合需求。因为不期望每分钟开关很多机器，所以我们还可以增加租约时间。我们把默认值设置为 6 个小时，最大租用时间值设置为 24 个小时。

```
default-lease-time 21600;
max-lease-time 86400;
```

完成之后，我们可以添加一个子网，DHCP 服务器在子网上分发租用的地址。

```
subnet 192.168.0.0 netmask 255.255.255.0 {
}
```

当 DHCP 服务器开启并自动把每个子网声明分配给正确的网络接口时，它会检查分配给本地网络接口的网络地址。

现在我们可以在这个配置块内添加子网专用选项。我们从定义用作默认路由的那个地址和用作域名服务器的那个主机的选项开始。

```
subnet 192.168.0.0 netmask 255.255.255.0 {
    option routers 192.168.0.254;
    option domain-name "example.com";
    option domain-name-servers 192.168.0.1;
    option broadcast-address 192.168.0.255;
}
```

这里定义了当客户机请求租约时应该发送给它的那些网络设置。`router` 选项指定了由客户机使用的默认网关。`domain-name` 选项不言自明。在 `domain-name-servers` 选项里，我们可以添加 1 个或更多 DNS 服务器地址，地址之间用空格分开。广播地址是网络上用来给同一网络范围上的所有主机发送请求的特殊地址，通过 `broadcast-address` 选项指定它。

不过还没有指定允许 DHCP 服务器分发的地址。我们使用 `range` 指令完成此事。

```
subnet 192.168.0.0 netmask 255.255.255.0 {
    option routers 192.168.0.254;
```



```
option domain-name "example.com";
option domain-name-servers 192.168.0.1;
option broadcast-address 192.168.0.255;
option subnet-mask 255.255.255.0;
range 192.168.0.101 192.168.0.200;
}
```

它告诉服务器如果一个客户机请求租约，它可以分配 192.168.0.101 到 192.168.0.200 之间的任一地址。这里没有指定完整的网络范围，从而保留了一些地址以便手动分配给服务器和其他主机。

现在剩下要做的事就是告诉 DHCP 服务器它应该监听哪些网络接口。如果没有做这件事，它就不会开始。在 Ubuntu 上，我们可以通过编辑 `/etc/default/dhcp3-server` 文件并把希望服务器监听的每个接口添加到 `INTERFACES` 变量来指定它。

```
INTERFACES="eth0"
```

在 Red Hat 上，我们通过编辑 `/etc/sysconfig/dhcpd` 文件并把每个接口添加到 `DHCPDARGS` 变量来完成此事。

```
DHCPDARGS="eth0"
```

保存此文件，然后在 Ubuntu 上使用 `sudo invoke-rc.d dhcp3-server start` 或者在 Red Hat 上使用 `sudo service dhcpd start` 开启服务器。

当服务器给特定的客户机分配一个租约时，它把客户机的 MAC 地址和所分配的租约记录到一个文件中。在重新连接时，它通常试图给客户机重新分配同一个地址，即使已经过了 `max-lease-time` 所规定的时间。当然，如果这个地址因为已分配给另一个客户机而不可用，服务器就要分发其他地址。

9.3.2 静态租约分配

有时你希望能够把同一个 IP 地址分配给某个主机或设备——如，一个网络打印机或者一个保存开发网站的工作站。你可以在客户机上手动编辑配置，但是那意味着要登入客户机做网络配置的修改。

DHCP 允许通过使用主机的 MAC 地址匹配来为它分配同一个 IP 地址。如果利用了这一点，你就可以修改任何主机的地址分配方法，只需编辑 `dhcpd.conf` 文件，重启 DHCP 服务，然后等待主机更新它的租约。

记得通过运行 `ifconfig` 命令你可以获得一个主机的 MAC 地址。你还可以运行 `arp` 命令列出本地网络上的 IP 地址和相关联的 MAC 地址。

这些配置指令都在 `dhcpd.conf` 文件里的 `subnet` 块内。我们开始先定义一个组，可以给它起任何你喜欢的名字；这里选择了“static”。

```
subnet ... {
    group "static" {
    }
}
```


我们接下来添加一个主机定义。每个主机被定义在组定义内属于它自己的块中。hardware ethernet 选项所指定的 MAC 地址会分配 fixed-address 选项所指定的地址。

■注：我们可以通过 ifconfig 命令找出网络接口的 MAC 地址。

这个选项可以包含一个 IP 地址或者一个可解析的 FQDN。之所以使用 FQDN，是因为 DNS 工作得很好。这还意味着如果想修改分配给这台主机的 IP 地址，但不想修改它的主机名，那么只需更新 DNS 区域文件，而不需要修改 DHCP 服务器。

```
subnet ... {
    group "static" {
        host au-mel-ubuntu-2{
            hardware ethernet 00:16:3E:15:3C:C2;
            fixed-address au-mel-ubuntu-2.example.com;
        }
    }
}
```

把 use-host-decl-names 标志设置为 on，确保了在主机块上所设置的名字——在我们的例子中是指 au-mel-ubuntu-2——将作为它应该使用的名字发送给 DHCP 客户机。如果不设置它，就必须为以这样方式定义的每台静态主机添加一个具体的主机名选项。因为是在组内定义的，所以它不会应用到组外。

```
subnet ... {
    group "static" {
        use-host-decl-names on;
        host au-mel-ubuntu-2{
            hardware ethernet 00:16:3E:15:3C:C2;
            fixed-address au-mel-ubuntu-2.example.com;
        }
    }
}
```

最后，我们希望确保为静态 DHCP 租约所使用的地址绝不会被分配给 DHCP 服务器所不知道的客户机。为此，我们可以通过定义地址池预留 100 个地址中的一些地址用于此目的。首先我们为 DHCP 知道的主机定义一个地址池。这些地址池的定义也是在子网块内进行的。

```
subnet ... {
    ...
    pool {
        range 192.168.0.101 192.168.0.150;
        deny unknown clients
    }
}
```

它为需要静态分配地址的主机预留了 50 个地址。接下来我们为其他所有主机定义一个地址池。对于该地址池，还忽略了租约时间，因为来访机器通常不会一整天都需要一个地址。

```
subnet ... {
    ...
    pool {
```



```

        range 192.168.0.101 192.168.0.150;
        deny unknown clients
    }
    pool {
        range 192.168.0.151 192.168.0.200;
        allow unknown clients;
        default-lease-time 7200;
        max-lease-time 21600;
    }
}

```

原来的 IP 地址范围分成了两部分。为了确保服务器不会认为被允许两次分配同一范围，我们要注释掉文件顶部的初始语句。

```

subnet ... {
    ...
    // range 192.168.0.101 192.168.0.200;
    ...
}

```

现在我们可以重启 DHCP 服务器了。剩下要做的事就是确保 DHCP 请求能够通过防火墙到达服务器。DHCP 客户机还没有分配 IP 地址，因此它不能给某个网络地址发送数据包。

而它所做的是给地址 255.255.255.255 的端口 67 广播一个 UDP 包，这个地址是 0.0.0.0 网络的广播地址。DHCP 服务器知道要监听这些包，而且如果接收到一个包，它会回应。因此我们需要配置 DHCP 服务器主机上的防火墙以接受在任何地址的端口 67 上的包。

```
$ sudo /sbin/iptables -t filter -A Firewall-eth0-INPUT -p udp --dport 67 -j ACCEPT
```

现在我们有了一个 DHCP 服务器配置。在此配置中，它给所规定的主机分配指定的预留 IP 地址，并给其他主机提供另一个地址范围。通过让这些主机使用预置的 IP 地址组，我们还可以设置防火墙规则并修改服务器配置，根据主机连接所用的地址接受或者拒绝访问，从而控制它们的访问。

9.3.3 动态 DNS 更新

可能你还希望给特定主机分配固定的 DNS 名，而不管 DHCP 服务器分配了什么 IP 地址。这样即使它们的地址变动了，也可以用名字指代机器。这通过设置由 DNS 和 DHCP 服务器两者共享的密钥完成。然后 DHCP 服务器会在它发出一个新租约时联系 DNS 服务器，如果需要的话，它会更新相关的 A 和 PTR 输入项。

1. 配置 DNS

在 Ubuntu 上，我们从生成密钥开始。为此我们将使用 `dnssec-keygen` 工具。我们指定带 `-a` 选项的 HMAC-MD5 算法，并用 `-b` 选项指定一个 512 位的密码长度。要使用的密钥名类型是 HOST，并用 `-n` 选项指定它。最后，我们传递想要使用的密钥名。

```
$ dnssec-keygen -a HMAC-MD5 -b 512 -n HOST dynamic-update-key
Kdynamic-update-key.+157+05884
```



```
$ ls -l
-rw----- 1 jsmith jsmith 127 Feb 10 22:16 Kdynamic-update-key.+157+05884.key
-rw----- 1 jsmith jsmith 145 Feb 10 22:16 Kdynamic-update-key.+157+05884.private
```

产生的密钥存储在 `Kdynamic-update-key.+157+05884.private` 文件里。

```
$ cat Kdynamic-update-key.+157+05884.private
Private-key-format: v1.2
Algorithm: 157 (HMAC_MD5)
Key: 3PDRnypPtzJqpbQvbw/B7bhPuHqpUe0Sdi95Z4Ez/IzhS6ldzcK6MJ6CdFHkkegpTN1kmXOM6GggRNE
24aPmOw==
```

注意，密钥文件名会因为密钥 ID（即第 2 组数据）的不同而不同。目前的任务不需要 `Kdynamic-update-key.+157+05884.key` 文件。

现在我们要向 `named` 和 `dhcpcd` 两者的配置中添加此密钥。从前者开始，在 `Ubuntu` 上把密钥定义添加到 `/etc/bind/named.local.conf` 文件，如列表 9-28 所示。

列表 9-28 往 bind9 配置里添加密钥

```
key dynamic-update-key{
    algorithm HMAC-MD5;
    secret "3PDRnypPtzJqpbQvbw/B7bhPuHqpUe0Sdi95Z4Ez/IzhS6ldzcK6MJ6CdFHkkegp
TN1kmXOM6GggRNE24aPmOw==";
};
```

DNS 知道这个密钥，但是仍需告诉它允许提供此密码的客户机更新特定的区域。为此，我们通过修改区域定义把 `allow-update` 指令包含进来，如列表 9-29 所示。

列表 9-29 允许使用一个密钥进行区域更新

```
zone "example.com" {
    type master;
    notify yes;
    allow-transfer {
        192.168.0.1;
    };
    file "example.com.db";
    allow-update {
        key dynamic-update-key;
    };
};

zone "0.168.192.in-addr.arpa" {
    type master;
    notify yes;
    allow-transfer {
        192.168.0.1;
    };
    file "192.168.0.db";
    allow-update {
        key dynamic-update-key;
    };
};
```

然后我们通过 `sudo rndc reload` 命令重新载入域名服务器的配置。

在 Red Hat 上，你可以使用 system-config-bind 工具添加密钥。我们通过“系统”→“管理”→“服务器设置”→“域名系统”启动它。然后突出显示“DNS 服务器”并选择“新建”→“安全密钥”。

这里需要做的就是输入希望给此密钥起的名字。当单击“新建”按钮时，一个密钥就自动生成了。之后，单击“确定”把此密钥添加到域名服务器配置中，如图 9-18 所示。

接下来你要修改区域定义以允许此密钥执行更新。在列表中选择区域（在我们的例子中就是 example.com），然后单击“属性”按钮。在选项窗口，突出显示“所有选项”列表里的“allow-update”，然后单击“+”把它添加到“当前选项”列表，如图 9-19 所示。



图 9-18 创建一个 DNS 安全密钥



图 9-19 修改区域选项

突出显示“当前选项”列表里的“allow-update”选项，在这里你可以修改它的属性。这次按从右到左的顺序操作。首先，往下滚到“新列表元素”列表，这样就可以突出显示“命名的密钥”。这样做会显示“编辑列表元素”列表中的可用密钥。图 9-20 突出显示了新的“dynamic-update-key”。

当单击“添加”时，“dynamic-update-key”就被添加到“地址匹配列表”了，如图 9-21 所示。

单击“确定”关闭此窗口。这时会得到一个警告，表明如果区域文件保留在/var/named 目录，名称服务器就不会更新它，如图 9-22 所示。

这是一个安全考虑。如果攻击者能够损害 DNS 服务器，这可以阻止他们修改磁盘上的区域。当然，如果 DHCP 服务器要更新一个区域，就需要写区域文件，因此你可以把它改存到 /var/named/slaves 目录，此目录对 DNS 服务器可写。单击“是”接受改动。

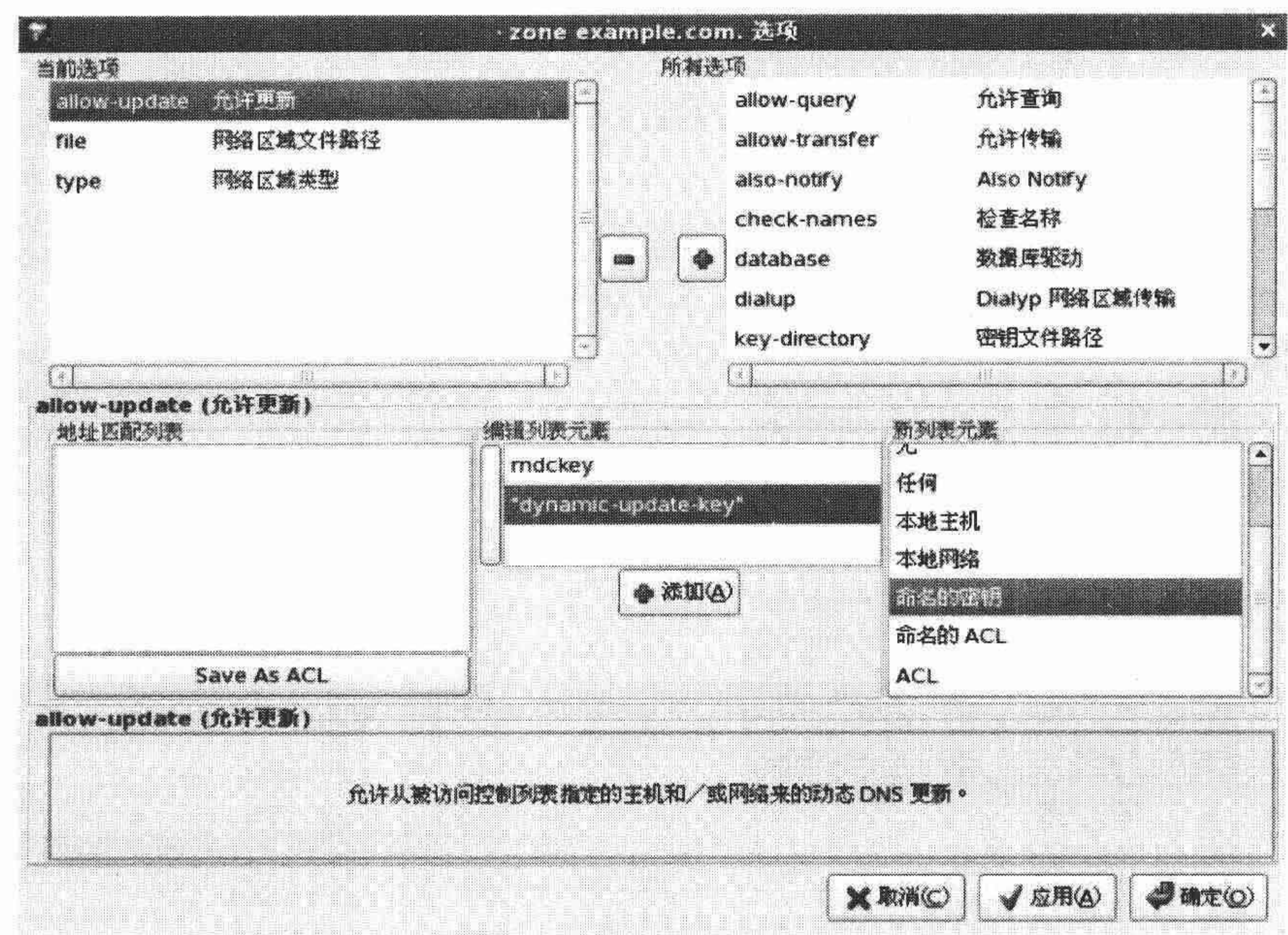


图 9-20 允许使用密钥更新区域



图 9-21 区域配置完成

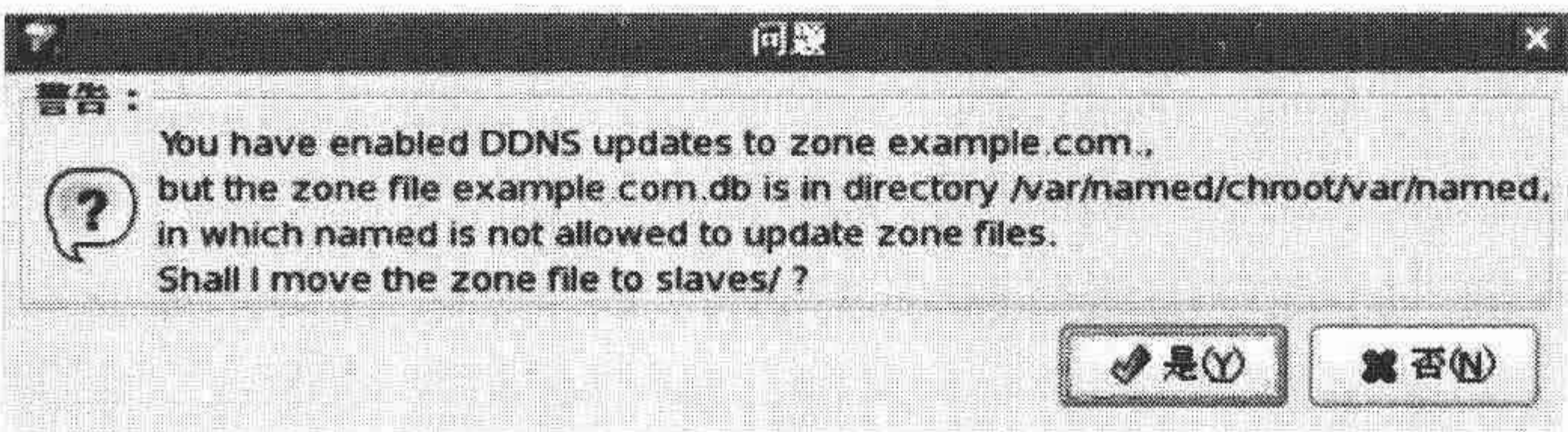


图 9-22 把区域文件移动到一个可写的目录

现在你需要对逆向区域做完全相同的事情。打开它的属性窗口，为“dynamic-update-key”添加“allow-updates”权限，正如为正向区域所做的那样。当接受了这个改动，它会再次询问是否想把区域文件转移到 slaves 目录，应该接受。

通过单击“保存”并选择备份和替换已存在的配置文件，把已更新的区域文件写入磁盘，如图 9-23 所示。

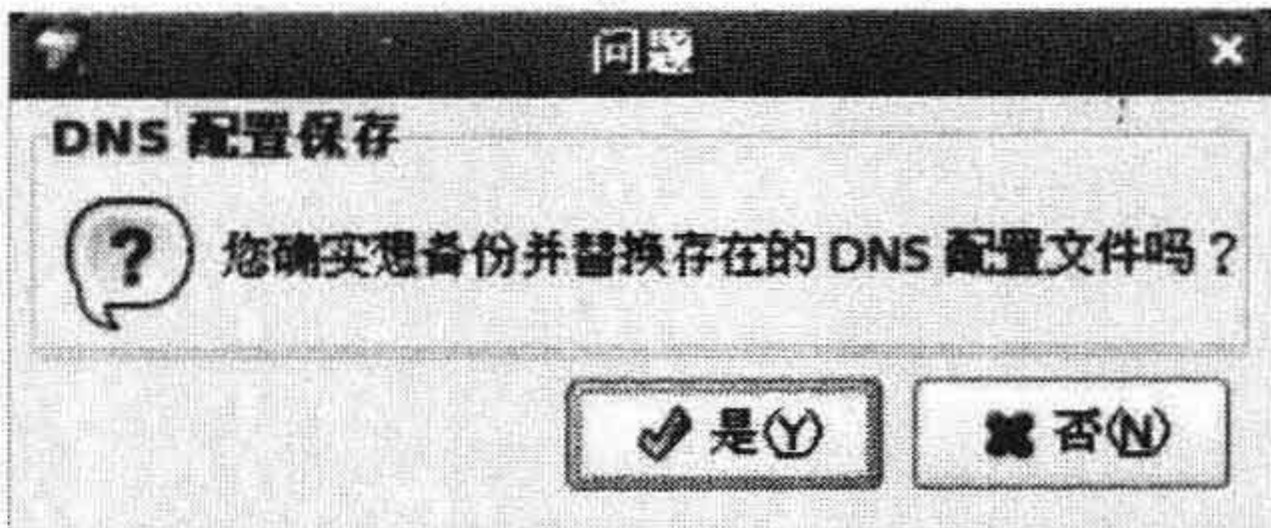


图 9-23 接受并保存改动

提示：你还可以在主 DNS 服务器和从 DNS 服务器之间使用基于密钥的身份验证机制。

2. 配置 DHCP

下一步是把此密钥告诉给 DHCP 服务器，并配置它使之在分发新的租约时给 named 发送 DNS 更新请求。开始我们先把 ddns-update-style 变量从 none 改为 interim。我们还希望对这些动态 DNS 输入项采用相当低的 TTL，从而使它们在主机从网络消失之后不起作用。我们指定为 1 小时。

```
ddns-update-style interim;
ddns-ttl 3600;
```

在它下面添加密钥定义，这仍然属于全局配置部分。重要的是，我们要使用与域名服务器上所用完全相同的密钥名，否则就不能更新。

```
key dynamic-update-key {
    algorithm hmac-md5;
    secret
    "3PDRnypPtzJqpbQvbw/B7bhPuHqpUe0Sdi95Z4Ez/IzhS6ldzcK6MJ6CdFHkkegpTN1kmXOM6Ggg
    RNE24aPmOw==";
}
```

最后我们要告诉 DHCP 服务器希望在正向和逆向区域上执行动态更新。我们需要在配置文件的全局部分为每个区域添加一个区域定义。我们还要指定哪些密钥应该用于更新以及这些更新要发送给哪些 DNS 服务器，如列表 9-30 所示。

列表 9-30 往 dhcpd.conf 里添加区域更新定义

```
zone 0.168.192.in-addr.arpa. {
    key dynamic-update-key;
    primary 192.168.0.1;
}

zone example.com. {
    key dynamic-update-key;
    primary 192.168.0.1;
}
```

我们最好重新启动 DHCP 服务器，然后服务器的配置就完成了。现在剩下要做的就是告诉 DHCP 客户机在它们请求租约时给服务器发送一个主机名字符串。然后这个主机名字符串被用来为 DNS 输入项创建 FQDN。

为了设置它，我们编辑客户机上的文件/etc/dhclient.conf 并添加 send host-name 选项。在一个希望命名为 au-mel-rhel-2.example.com 的主机上，我们添加下面的命令。

```
send host-name "au-mel-rhel-2";
```

■注：/etc/dhclient.conf 文件里的配置应用于所有网络接口。你可以使用/etc/dhclient-eth0.conf 文件改为只把它应用于第一个接口。

然后我们可以运行 dhclient 程序来更新地址租约，如列表 9-31 所示。如果配置主机时使用 DHCP，它也会在启动时自动运行。如果是这样的话，并且希望手动更新租约，那么我们首先应该杀掉正在运行的 dhclient 进程。

列表 9-31 用 dhclient 获得租约

```
$ sudo dhclient eth0
Internet Systems Consortium DHCP Client V3.0.5-RedHat
Copyright 2004-2006 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/sw/dhcp/

Listening on LPF/eth1/00:0c:29:7b:b1:77
Sending on LPF/eth1/00:0c:29:7b:b1:77
Sending on Socket/fallback
DHCPREQUEST on eth0 to 255.255.255.255 port 67
DHCPACK from 192.168.0.1
bound to 192.168.0.200 -- renewal in 7181 seconds.
```

我们可以检查服务器的系统日志看看发生了什么。列表 9-32 中摘录了一部分内容。在 Red Hat 上这个日志文件是/var/log/messages，在 Ubuntu 上它是/var/log/syslog。第 18 章会说明如何把特定的日志消息重导向不同的文件。

列表 9-32 DHCP 服务器日志

```
Feb 11 11:23:15 au-mel-ubuntu-1 dhcpd: DHCPDISCOVER from 00:0c:29:7b:b1:77 via eth0
Feb 11 11:23:16 au-mel-ubuntu-1 dhcpd: DHCPOFFER on 192.168.0.200 to
00:0c:29:7b:b1:77 (au-mel-rhel-2) via eth0
Feb 11 11:23:16 au-mel-ubuntu-1 named[5187]: client 192.168.0.1#46749: updating
zone 'example.com/IN': adding an RR at 'au-mel-rhel-2.example.com' A
Feb 11 11:23:16 au-mel-ubuntu-1 named[5187]: client 192.168.0.1#46749: updating
zone 'example.com/IN': adding an RR at 'au-mel-rhel-2.example.com' TXT
Feb 11 11:23:16 au-mel-ubuntu-1 named[5187]: journal file example.com.db.jnl does
not exist, creating it
Feb 11 11:23:16 au-mel-ubuntu-1 dhcpd: Added new forward map from
au-mel-rhel-2.example.com to 192.168.0.200
Feb 11 11:23:16 au-mel-ubuntu-1 named[5187]: zone example.com/IN:sending
notifies (serial 2009020102)
Feb 11 11:23:16 au-mel-ubuntu-1 named[5187]: client 192.168.0.1#58073: updating zone
'0.168.192.in-addr.arpa/IN': deleting rrset at '200.0.168.192.in-addr.arpa' PTR
Feb 11 11:23:16 au-mel-ubuntu-1 named[5187]: client 192.168.0.1#58073: updating zone
'0.168.192.in-addr.arpa/IN': adding an RR at '200.0.168.192.in-addr.arpa' PTR
Feb 11 11:23:16 au-mel-ubuntu-1 named[5187]: journal file 192.168.0.db.jnl does
not exist, creating it
Feb 11 11:23:16 au-mel-ubuntu-1 dhcpd: added reverse map from
200.0.168.192.in-addr.arpa. to au-mel-rhel-2.example.com
```



```
Feb 11 11:23:16 au-mel-ubuntu-1 dhcpd: DHCPREQUEST for 192.168.0.200 (192.168.0.1)
    from 00:0c:29:7b:b1:77 (au-mel-rhel-2) via eth0
Feb 11 11:23:16 au-mel-ubuntu-1 dhcpd: DHCPACK on 192.168.0.200 to
    00:0c:29:7b:b1:77 (au-mel-rhel-2) via eth0
```

可以看到，服务器接收了一个来自 MAC 地址为 00:0c:29:7b:b1:77 的主机的请求。然后它给这个主机提供了 192.168.0.200 的地址，并被告知该主机名为 au-mel-rhel-2。接下来可以看到域名服务器为 au-mel-rhel-2.example.com 的 FQDN 添加了一个 A 记录和一个 TXT 记录。TXT 输入项包含一个校验和，用于追踪 DHCP 服务器是否创建了一个 DNS 输入项。如果它不存在，服务器会修改或移除相关联的 A 输入项。

然后对区域的改动被写入一个早前所创建的与此区域文件相关联的日志文件。实际的区域文件本身并没有被修改。正向区域更新了之后，它给所有为此区域配置的从服务器发送一个通知。如果有从服务器的话，这将触发从服务器传递来自主服务器的区域文件。

接下来，对逆向区域重复同样的过程。完成后，DHCP 服务器允许客户机获得它所提供的租约，并更新它的内部租约文件。我们可以通过 host 命令执行一个查找，快速复检这些新的 DNS 输入项。

```
$ host 192.168.0.200
200.0.168.192.in-addr.arpa domain name pointer au-mel-rhel-2.example.com.
$ host au-mel-rhel-2.example.com.
au-mel-rhel-2.example.com has address 192.168.0.200
```

两次查找都正常，因此对于那些同样希望在 DNS 服务器里动态更新的网络上的其他主机，现在我们可以配置了。

9.3.4 手动修改 DNS 输入项

因为这些动态更新使用一个日志文件，所以如果想手动修改 DNS 输入项，你需要执行额外的一个步骤。如果只是修改区域文件，这些改动会被忽略，因为日志文件里的数据会取代它。

通过在开始编辑一个区域之前发出 `sudo rndc freeze` 命令，可以告诉 DNS 服务器在编辑过程中要锁定区域日志文件并拒绝任何的动态改动。编辑完区域文件后，你可以再通过 `sudo rndc unfreeze` 命令解锁此区域，以允许动态更新。

补充知识：Internet 超级服务器

不是所有的网络服务都需要一直运行。对于其中一些服务，更方便的方式是在特定端口号上监听流量，在需要时才开启服务。一旦客户端断开了，服务也会关闭。不是所有的服务都支持这种方式，但是如果它们支持的话，这是一个很好的节约系统资源的方式。

“监控程序”功能是由 Internet 超级服务器提供的。此软件功能最丰富、最新的版本由 xinetd 软件包提供。

xinetd 所管理的每个服务在 `/etc/xinetd.d` 目录里都有它自己的配置文件片段。默认情况下，它会为一些非常基本的服务安装配置文件，如只是重复用户所发送字符的 echo。不过，如果不编辑这些服务的配置文件，它们不会被启用。

第 19 章将说明如何往 xinetd 添加一个服务。

9.4 Secure Shell

迄今为止，其实只是用 SSH 从工作站连接到服务器，以改动配置或添加新软件。现在我们要说明如何发挥 SSH 的最大效用。我们将设置基于密钥的身份验证功能，使用 ssh 在主机之间复制文件，并利用隧道通过防火墙访问远程服务。

当通过 SSH 连接到一个主机时，它要求输入密码。如果一天输入一次，这很好，但是如果经常连接到远程主机，它可能变得耗时，特别是当密码安全又长的时候更是如此。

SSH 允许改用基于密钥的身份验证功能。为了利用它，你要创建公钥和私钥，然后把公钥复制给要连接的远程服务器。在连接时，远程主机会验证你是否拥有属于那个主机上公钥组件的私钥。如果有，你就可以通过身份验证。

注：公钥和私钥用于身份证明。连接加密由 SSH 主机密钥提供，这个密钥是在电话服务时产生的。

9.4.1 创建和分配密钥

我们从使用 ssh-keygen 工具创建一个公钥/私钥对开始。我们可以定义密钥类型（支持两种加密算法）和密钥位数，以及要使用的输出文件名。对于后者使用默认值，对于前者用 -t 选项指定 RSA 算法，使用 -b 选项指定密钥长度为 2048 位，如列表 9-33 所示。

列表 9-33 产生一个新 SSH 密钥对

```
$ ssh-keygen -t rsa -b 2048
Generating public/private rsa key pair.
Enter file in which to save the key (/home/jsmith/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/jsmith/.ssh/id_rsa.
Your public key has been saved in /home/jsmith/.ssh/id_rsa.pub.
The key fingerprint is:
8c:57:6d:26:ba:6c:47:98:55:25:ce:4d:50:51:f8:85 jsmith@au-mel-ubuntu-1.example.com
```

重要的是要为私钥增加一个密码口令，因为如果没有的话，任何得到该私钥的人都可以使用它登录到（不需要密码）包含对应公钥的任何主机。

既然有了一个密钥对，我们就可以把公钥部分复制给远程主机。我们需要把公钥保存在主目录下 .ssh 目录中叫作 authorized_keys 的文件里，以便能够使用它登入。我们可以手动把密钥添加到那个文件，或者使用 ssh-copy-id 工具来做，如列表 9-34 所示。

列表 9-34 把一个 SSH 公钥复制到远程主机

```
$ ssh-copy-id au-mel-rhel-1.example.com
The authenticity of host 'au-mel-rhel-1.example.com (192.168.0.1)' can't be
established.
```



```

RSA key fingerprint is 67:e3:50:bf:8c:2c:a0:d5:0c:e9:fc:26:3f:9f:ea:0e.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'au-mel-rhel-1.example.com,192.168.0.1' (RSA) to the
list of known hosts.
jsmith@au-mel-rhel-1.example.com's password:

```

现在我们使用下面的命令尝试登入那台机器。

```
$ ssh au-mel-rhel-1.example.com
```

因为还没有从登入的主机连接到 `au-mel-rhel-1`，所以它会提示接受远程 SSH 主机密钥。识别该密钥的独特指纹已经印好，可以直观地验证它是否与远程主机上的密钥匹配。

■注：为了获得一个主机密钥指纹，你可以使用 `ssh-keygen` 工具。本例中使用 `ssh-keygen -l -f /etc/ssh/ssh_host_rsa_key.pub` 来获得主机 RSA 密钥的指纹。

注意，SSH 假定在远程主机上的用户名与作为本地用户登入的用户名相同。如果不是这样，我们可以把密钥复制到 `username@remotehost`。

接下来，它提示输入登录密码，因为密钥还没有列在远程主机的适当的文件里。一旦通过身份验证，`ssh-copy-id` 就会把公钥追加到适当的文件并要求测试它。为此，我们登入到远程主机，如列表 9-35 所示。

列表 9-35 使用 SSH 密钥登录

```

$ ssh au-mel-rhel-1.example.com
Enter passphrase for key '/home/jsmith/.ssh/id_rsa':
Last login: Tue Feb 10 15:14:42 2009 from au-mel-ubuntu-1.example.com
[jsmith@au-mel-rhel-1 ~]$

```

这次，它没有要求输入 `au-mel-rhel-1` 的登录密码，这正是我们所希望的。我们现在可以检查 `au-mel-rhel-1` 上的 `.ssh/authorized_keys` 文件，以确保没有添加其他意外的密钥。

9.4.2 使用 SSH 代理

无论如何，我们还是必须输入在 SSH 私钥上设置的密码。如果每次想连接到远程主机时都必须这么做，那么设置基于密钥的身份验证功能就失去意义了。进入 SSH 代理，这个小后台程序在内存里保管着解锁的 SSH 私钥。一旦开启了它，我们就可以解锁一个或更多私钥并把它们添加到代理中。然后 SSH 可以使用代理提供一个私钥，并向远程主机证明身份。

把这个代理告诉给 SSH 的方法是设置两个环境变量：`SSH_AUTH_SOCK` 和 `SSH_AGENT_PID`。如果设置了它们，ssh 就可以和代理通信了。代理程序在启动时输出 shell 代码以设置这些变量，如列表 9-36 所示。

列表 9-36 开启 ssh-agent

```

$ ssh-agent
SSH_AUTH_SOCK=/tmp/ssh-SZGGF11534/agent.11534; export SSH_AUTH_SOCK;
SSH_AGENT_PID=11535; export SSH_AGENT_PID;
echo Agent pid 11535;

```


如果接着把这些行粘贴到 shell 里，这些变量就设置好了。

```
$ SSH_AUTH_SOCKET=/tmp/ssh-SZGGF11534/agent.11534; export SSH_AUTH_SOCKET;
$ SSH_AGENT_PID=11535; export SSH_AGENT_PID;
$ echo Agent pid 11535;
Agent pid 11535
```

还需要复制、粘贴这些行，有点麻烦，那么为了更容易一点，我们可改用 `eval shell` 函数。这个函数把传递给它的参数当作命令一样执行。首先，我们通过 `ssh-agent -k` 停止代理，然后重启它，一次性全部设置好那些环境变量。参数上的反引号会使 shell 把它当作一个命令执行。然后这个命令产生的输出由 `eval` 解释。

```
$ ssh-agent -k
unset SSH_AUTH_SOCKET;
unset SSH_AGENT_PID;
echo Agent pid 11535 killed;
$ eval `ssh-agent`
Agent pid 11541
```

现在需要做的就是解锁私钥并把它添加到代理中。

```
$ ssh-add
Enter passphrase for /home/jsmith/.ssh/id_rsa:
Identity added: /home/jsmith/.ssh/id_rsa (/home/jsmith/.ssh/id_rsa)
```

我们可以连接到包含匹配公钥的任何主机，而无需再输入任何密码。

```
$ ssh jsmith@au-mel-rhel-1
Last login: Tue Feb 10 15:17:19 2009 from au-mel-ubuntu-1.example.com
[jsmith@au-mel-rhel-1 ~]$
```

■提示：通过直接在 shell 里把 `SSH_AUTH_SOCKET` 和 `SSH_AGENT_PID` 变量设置为正确的值，你可以把正在使用代理这件事告诉给同一台主机上的多个 shell。

9.4.3 调整 SSH 配置

当所有的 SSH 服务器都在同一个端口监听并且在所有的主机上使用单个密钥对时，默认的服务器配置就很合适。如果不是这样的话（例如，端口 22 上的流量可能被防火墙控制，或者远程用户名对每一个主机都不一样），你可能就要稍微调整一下服务器或客户机的配置。

1. 基本的服务器配置

服务器的 SSH 从 `/etc/ssh/sshd_config` 文件读它的配置。默认情况下，它监听所有可用网络接口上的端口 22。你可以通过修改配置文件里的 `Port` 和 `ListenAddress` 选项修改它。

`Port` 选项带一个单独的参数，就是希望服务器监听的端口号。要让服务器监听多个端口，你可以添加另外的 `Port` 指令，一个指令设置一个端口号。

这也同样适用于 `ListenAddress` 指令。只要不存在这样的指令，服务器就会监听所有的接口。如果添加了一个，它就只监听指定地址上所规定的端口。你可以通过添加另外的

`ListenAddress` 指令让它在多个地址上监听。

例如，为了让堡垒主机上的 SSH 服务器只监听内部网络接口上的端口 22 和 2022，我们可以把这些指令添加到配置文件中。

```
Port 22
Port 2022
ListenAddress 192.168.0.1
ListenAddress 19.168.1.1
```

现在我们可以 `Red Hat` 上通过 `sudo service sshd reload` 或者在 `Ubuntu` 上使用 `sudo invoke-rc.d ssh reload` 命令告诉服务器重新载入它的配置文件。这不会影响当前连接，因此你可以远程运行这个命令。

注：确保不要把 SSH 服务器重配置到你不能访问的程度！如果担心的话，配置更改后不要登出。我们首先尝试创建一个新的连接，以确保它仍能工作。

下面要介绍的另一个基本的服务器选项是为了在远程主机上更容易与 GUI 程序打交道而设计的。如果 `X11Forwarding` 选项设置为 `on`，并且在连接到这样的主机时把 `-X` 参数传递给了 SSH 客户机，你就可以运行任何图形程序，它们的窗口会显示在本地桌面上。这个功能利用了 X 窗口系统的客户端/服务器模式，它通过 SSH 连接把给远程主机上 X 服务器的连接尝试转发给本地主机的 X 服务器。

为了强制所有用户都使用基于密钥的身份验证功能，你可以在服务器配置文件里添加 `PasswordAuthentication no`。它会阻止所有人用密码登录。注意，在设置了这个选项的情况下，如果私钥丢失了，你将不能再次登入主机。

在 `man sshd_config` 帮助页面可以找到所有可用服务器配置选项的完整列表。

2. 客户端配置

SSH 客户端可以为主机上所有用户进行全局配置，也可以针对每个用户进行局部配置。全局配置文件是 `/etc/ssh/ssh_config`，每个用户的配置文件是用户主目录里的 `.ssh/config`。

最基本的客户端配置指令允许规定在连接到给定主机或所有主机时所使用的用户名和端口号。这些配置块每一个都是以 `Host` 指令开始，跟着是一个主机名或简写的别名（如果这部分只应用于单个主机）或者一个星号（如果它应用于所有主机）。

例如，通过往 `.ssh/config` 文件添加下面的代码片段，我们可以轻易地定制堡垒主机的连接选项。

```
Host gateway
    Hostname au-mel-rhel-1
    Port 2022
    User ataylor
```

这个配置在每次使用 `ssh gateway` 命令时使用。它告诉客户机连接到 `au-mel-rhel-1` 主机的端口 2022 上，并以 `ataylor` 用户身份登入。通过在客户端配置文件里添加这些选项，我们就不需要总在命令行里指定端口号和登录名。

类似地，当连接到远程主机时，我们可以告诉客户机使用一个不同的私钥文件，方法就

是使用 `IdentityFile` 指令添加它。通过使用命令 `ssh-keygen -t rsa -s 2048 -f .ssh/gateway-ataaylor`，在网关主机上生成了一个供 `ataaylor` 使用的密钥对。完成后，我们就可以告诉客户机使用此密钥连接堡垒主机。

```
Host gateway
  Hostname au-mel-rhel-1
  Port 2022
  User ataylor
  IdentityFile ~/.ssh/gateway-ataaylor
```

最后要涉及的选项是为了更方便地操作远程主机而设计的。首先，使用 `ForwardAgent yes` 选项可以告诉服务器它应该使用发起会话主机上的 SSH 代理进行身份验证。这使用户能够通过 SSH 从一个主机跳到另一个主机，而无需每次都在这些主机上输入密码来开启 SSH 代理。

因此不必为了开启 X 转发功能总给 `ssh` 添加 `-X` 参数，你最好在配置文件中针对单机打开它。对于每一个想在它上面远程运行 GUI 程序的主机，我们添加 `ForwardX11 yes` 指令以自动打开这个选项。

3. 隧道

你还可以在没有事先设置 VPN 的情况下使用 SSH 访问远程主机或网络上受保护的服务。如果两个站点共享相同的私有网络范围，VPN 就不会起作用，因为 VPN 两边的地址范围相同。在这种情况下，你可以使用 SSH 转发从本地主机到远程地址的连接或者相反的连接。这样的转发机制起到了一个单端口隧道的作用。

你可以通过命令行参数对每一个都这样做，或者在 `.ssh/config` 文件里为每个主机定义转发功能。例如，可以创建一个 SSH 隧道把到本地主机端口 8080 的连接转发到远程网络上某台机器的端口 80 上。这样你就可以通过浏览本地网络上的一个地址访问一个远程网站。创建本地转发功能的方式是，给 SSH 客户机传递 `-L` 选项并指定一个任选的本地地址以及一个强制的本地端口作为隧道的开始，然后指定一个远程主机和一个远程端口作为隧道的末端，所有项都用冒号分开。

```
$ ssh -L 8080:192.168.1.12:80 ataylor@192.168.1.
```

这个命令以 `ataaylor` 的用户身份把我们连接到主机 192.168.1.1，并设置了一个隧道，允许我们通过用网络浏览器访问 `http://localhost:8080` 来浏览主机 192.168.1.12 上的网站。这个连接可以通过 SSH 连接转发，在主机 192.168.1.12 上的网站服务器会看到一个来自地址 192.168.1.1 的连接。

■注：由于网站寄宿主机的方式不同，不是所有的网站都可以通过隧道访问。第 11 章将介绍这种基于域名的虚拟寄宿方式。

反之，通过使用 `-R` 选项创建一个远程转发功能，你可以为远程主机上的用户提供对本地网络上某个服务的访问。这个选项和 `-L` 选项的参数相同，但不同的是，这里指定一个任选的远程地址和在远程主机上监听的强制端口号，然后是本地地址和端口号作为隧道的终点。

为了让远程用户连接到本地网络上一个正常情况下不可访问的 SSH 服务器，我们可以在

端口 2022 上创建一个远程隧道，它把连接转发到本地网络上某个主机的端口 22。

```
$ ssh -R 192.168.1.1:2022:192.168.0.15:22 ataylor@192.168.1.1
```

以 ataylor 用户身份登入主机 192.168.1.1 之后，我们可以用 SSH 连接到本地主机的端口 2022 上，然后它会让我们登入到主机 192.168.0.15 上的 SSH。

为安全起见，隧道的开始永远只绑定到回送网络接口，这样，网络上其他主机的用户就不能使用这个隧道。通过把 GatewayPorts 指令添加到 SSH 服务器配置文件可以更改这种行为。这个选项只适用于转发的起始点，因此对于本地隧道，我们在本地主机上添加它；为了远程转发，我们在远程主机上添加它。

为了能够决定其他主机上的用户是否可以使用转发功能，把 GatewayPorts 选项设置为 clientspecified。如果没有为转发起始点指定一个 IP 地址，它将只对本地用户可访问，而如果指定一个可访问的地址，那么作为隧道起始点的同一网络上的任何用户都可以访问。

因为这需要打很多字，所以把常用的隧道定义在 SSH 客户端配置文件里就更容易些。这通过 LocalForward 和 RemoteForward 指令来执行。每一个指令都有 2 个参数：以冒号分隔的转发起始地址与端口，以及也是用冒号分隔的末端地址与端口。

我们可以把前面使用过的转发功能代码添加到客户端的配置文件。

```
Host gateway
  Hostname 192.168.1.1
  Port 22
  User ataylor
  IdentityFile ~/.ssh/gateway-ataylor
  LocalForward 8080 192.168.1.12:80
  RemoteForward 192.168.1.1:2022 192.168.0.15:22
```

最后，ForwardAgent yes 选项使 SSH 配置远程 shell 以便使用本地主机上的 SSH 代理进行身份验证。假如公钥在所有远程主机上都可用，你就可以在主机之间跳跃，而无需重新输入密码或者在每一个中间主机上开启新的 ssh-agent。这是一个极其有用的选项，因此最好把它添加到/etc/ssh/ssh_config 文件的全局部分，以便对所有用户都开启它。

9.4.4 执行快速又安全的文件传输任务

SSH 协议考虑的不仅仅是远程登录。使用它还可以在主机之间安全地传输文件。一种方式是使用 scp 命令，除了源文件或者目标文件可以添加远程用户名和主机名作为前缀，它的工作方式与 cp 一样，如列表 9-37 所示。

列表 9-37 使用 scp 向远程主机传输文件

```
$ scp data.txt jsmith@au-mel-rhel-1:/tmp
data.txt 100% 3072KB 3.0MB/s 00:00
```

因为前面已经把 SSH 公钥发送给 au-mel-rhel-1，所以 scp 可以使用 SSH 代理进行身份验证，而没有要求输入密码。现在登入 au-mel-rhel-1 主机，我们可以看到文件 data.txt 在/tmp 目录中。

我们还可以从一个远程主机复制到本地主机，方式是指定一个远程文件路径作为第 1 个

参数，一个本地文件或目录作为第2个参数。

```
$ scp jsmith@au-mel-rhel-1:/tmp/data.txt /tmp
data.txt                                100% 3072KB 3.0MB/s 00:00
```

我们甚至可以把文件或目录从一个远程主机复制到另一个远程主机，而无需登入它们。例如，在 `au-mel-ubuntu-1` 上我们可以运行下面的命令。

```
$ scp jsmith@au-mel-rhel-1:/tmp/data.txt ataylor@au-mel-rhel-2:/tmp
data.txt                                100% 3072KB 3.0MB/s 00:01
```

SSH 还提供了一个 FTP 协议的替代品。如果希望以交互的方式移动文件或目录，我们可以使用 `sftp` 命令，如列表 9-38 所示。如果 SSH 代理存在，这个命令也会使用它。

列表 9-38 使用 sftp

```
$ sftp jsmith@au-mel-rhel-1
Connecting to au-mel-rhel-1...
jsmith@au-mel-rhel-1's password:
sftp> cd /tmp
sftp> ls
data.txt  ssh-IWYooo5675
sftp> get data.txt
Fetching /tmp/data.txt to data.txt
/tmp/data.txt                                100% 3072KB 3.0MB/s 00:00
sftp> quit
```

提示：结合 SSH 端口转发功能，你还可以轻松地把文件复制到不可直接访问的主机。注意 `scp` 使用 `-P` 选项指定一个端口号，而 `ssh` 使用的是 `-p`。

9.5 小 结

在本章，我们学习了如 NTP、DNS 和 DHCP 这样的基本基础架构服务。我们还说明了如何连接到远程主机，如何使系统管理和维护更容易。现在你应该能做下面的事情。

- 为所有的主机设置并保持正确的时间。
- 为所有的主机创建正向和逆向 DNS 记录并把这些记录复制到多个 DNS 服务器。
- 设置 DHCP 使地址分配自动化并把它链接到 DNS 以自动更新相关的 DNS 记录。
- 使用 `ssh`、`scp` 和 `sftp` 轻松安全地在远程主机上工作以及传输文件。

下一章将介绍邮件服务，并教你运行自己的邮件服务器。

第 10 章 邮件服务

配置 Linux 主机最常见的理由之一是，通过如 Internet 消息控制协议（Internet Message Access Protocol, IMAP）和邮局协议（Post Office Protocol, POP3）这样的机制提供邮件服务，包括收发和检索电子邮件。本章将简要地讲解电子邮件是如何工作的，并介绍一个电子邮件方案的组件，包括以下几个。

- MTA，或者 Mail Transfer Agents：发送和接收电子邮件的服务器。
- MUA，或者 Mail User Agents：用户用来发送和接收电子邮件的客户端。
- MDA，或者 Mail Delivery Agents：帮助把电子邮件投递到邮箱的工具。

我们还会介绍执行这些功能的一些应用程序。

- Postfix：一个简单邮件传输协议（Simple Mail Transfer Protocol, SMTP）电子邮件服务器。
- Dovecot：一个 IMAP 和 POP3 服务器。

Postfix 电子邮件服务器允许用户发送和接收来自内部用户或外部（如 Internet）的电子邮件。Dovecot 服务器提供 IMAP 和 POP3 后台程序。对于用户从位于电子邮件服务器的邮箱检索邮件来说，IMAP 和 POP3 是两种不同的方式（我们会解释不同之处以及为什么使用这个而不是另一个）。

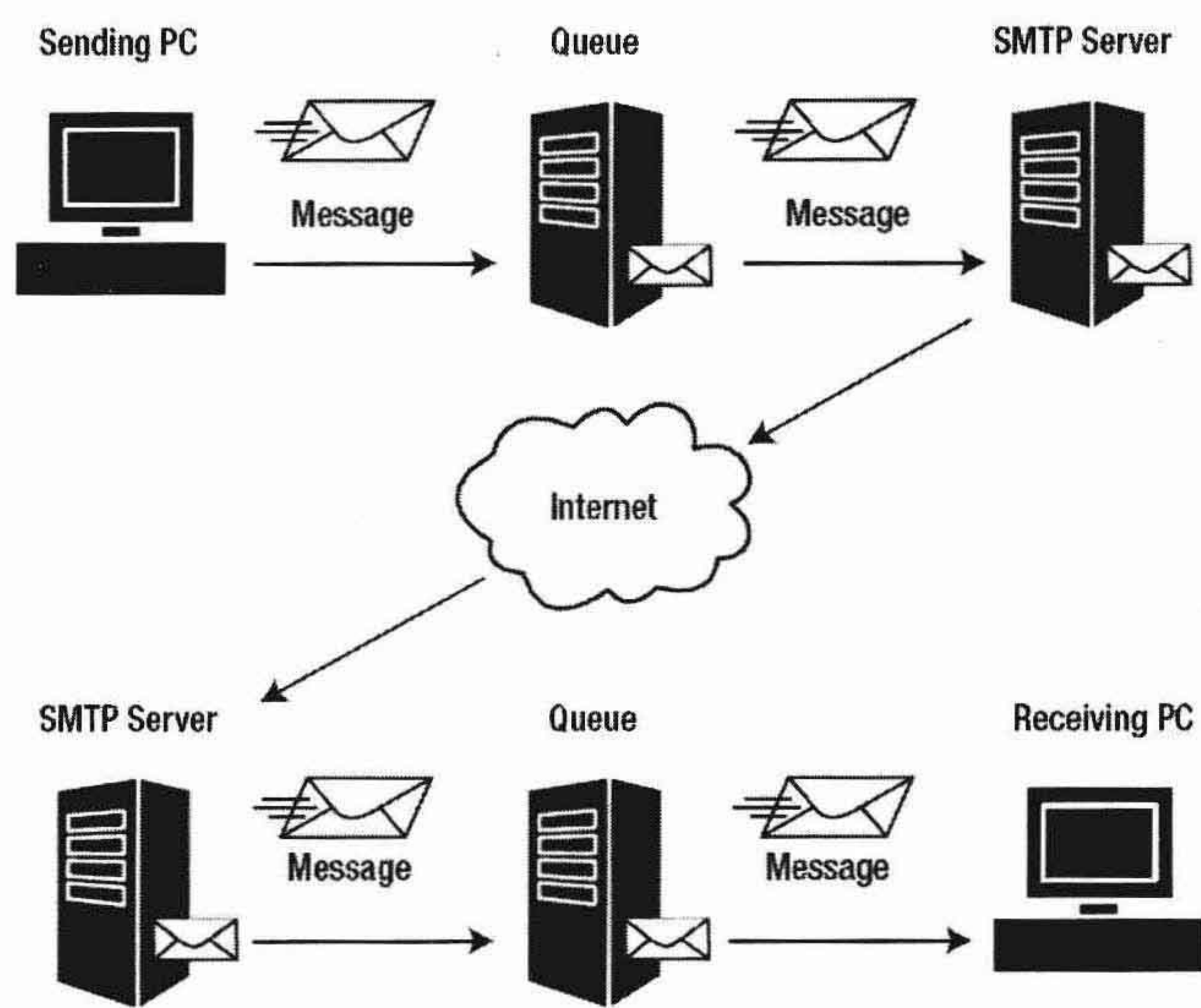
我们还将说明如何保护用户免受未经同意的邮件或垃圾邮件以及病毒的侵害。

重要的是，我们要理解本章所阐述的基本邮件服务——发送、接收和管理电子邮件。这不包括日程表、日历、待处理任务列表之类的服务，或者像 Microsoft Exchange、Zimbra 和 OpenExchange 那样的协作程序组。第 15 章会介绍开源协作程序组，除了这些其他功能外，它也包括电子邮件功能。如果觉得需要那些功能，你也许想跳到那一章。不过，我们强烈推荐阅读本章，从而对电子邮件的工作方式有一个了解。

10.1 电子邮件是如何工作的

电子邮件在个人和商务通信中已经无所不在。在许多情况下，它作为通信工具已经完全取代了邮政传真服务。大部分人除了编写电子邮件和单击“发送”按钮外，其他的都不必担

心。可是，为了运行自己的邮件服务器，我们需要了解电子邮件的内部工作方式。图 10-1 显示了典型的电子邮件生命周期。



Sending PC: 发送机; Message: 消息; Queue: 队列; SMTP Server: SMTP 服务器; Receiving PC: 接收机
图 10-1 电子邮件的生命周期

10.1.1 发送一封电子邮件时发生了什么事?

电子邮件基于一个叫作 SMTP 的协议（定义在 RFC 5321: <http://www.ietf.org/rfc/rfc5321.txt>）。每封电子邮件都有一连串的报头，以告诉邮件服务器如何处理它以及把它发往何处。因此当用户创建新的电子邮件时，要给邮件写上地址。他们添加一个收件人（或者 To 字段），而且可能把邮件抄送或者匿名发送给其他人。然后发送这封电子邮件。

用户的电子邮件客户端是用服务器配置的，现在客户端联系这个服务器（在 TCP 端口 25）并说“你好！我有从此人到这些人的邮件——请把它发送给他们！”事实上，客户端不会确切地说“你好”，它说的东西非常类似——一个叫作 EHLO（或者更少见的 HELO）的命令。

注：电子邮件服务器有很多名字，可以叫 SMTP 服务器或 SMTP 后台程序，但它们的专有名词是 Mail Transfer Agent。按照这种命名方式，发送电子邮件的客户端被称作 Mail User Agent，这已为大家所知。本章稍后还会介绍另外一个用来往用户邮箱投递电子邮件的叫作 Mail Delivery Agent 的组件。

EHLO 是一个 SMTP 命令，它是 SMTP “语言”的一部分。命令是 SMTP 客户端和服务 器（或者 MUA 和 MTA）互相沟通的方式。最初，SMTP 的语言里只有大约 10 个词或命令。在最近的时间里，SMTP 的加强版，叫作扩展的 SMTP（Extended SMTP, ESMTP）得以创建，它给这种语言增加了更多有用的命令，提供如身份验证和加密之类的功能。

■注：前面提到的 HELO 命令是 EHLO 命令的老版本。它已经很少用了，但正规的邮件服务器应该支持它，以防客户端是老版本而不认识 EHLO 的情况。

在本例中，客户端正在向服务器表明身份，从列表 10-1 可以看到用 SMTP “语言” 进行的一个简单会话。

列表 10-1 一个简单的 SMTP 会话

```
220 mail.example.com ESMTP Server
EHLO client.example.com
250 mail.example.com Hello client.example.com [192.168.0.100], pleased to meet you
```

在列表 10-1 的第一行，服务器告诉客户端它是 mail.example.com，并且是一个支持扩展的 SMTP（或 ESMTP）的 SMTP 服务器。

■注：你可以看到这行开头的数字是 2xx 的格式。它是客户端和服务器交换成功或失败信息的回应代码。如果这行开头数字是 5xx，客户端和服务器就知道这是某种错误的代码。稍后你会看到更多这样的数字。

在下面一行，客户端向服务器问好并告诉它自己是 client.example.com（客户端一般用完整的域名表示自己）。最后，服务器回应确认了此连接，并回复问候。

接下来，真正的电子邮件被发送给服务器。开始先发送发送者和收件人资料，这通常称为信封。

```
MAIL FROM: <ataylor@example.com>
250 ataylor@example.com... Sender ok
```

在这里发送者 ataylor@example.com 使用了 MAIL FROM 命令。服务器验证后返回 250 的回应代码，表示允许发送者提交邮件。这个接受的决定可以基于许多标准，包括电子邮件地址构造要正确，本章稍后将介绍身份验证之类的机制。

接下来，服务器期待 RCPT TO 命令，或者这封电子邮件要发送给的那个人。

```
RCPT TO: <jsmith@example.com>
250 jsmith@example.com... Recipient ok (will queue)
```

同样，这次接受该地址也是基于如邮件地址格式正确之类的标准。

在这个简单例子中，接下来需要的是邮件的正文。客户端向服务器发送一个 DATA 命令。

```
DATA
354 Enter mail, end with "." on a line by itself
Message-ID:
Date: Mon, 17 Aug 2009 12:29:26 +1100
From: Anne Taylor ataylor@example.com
To: John Smith jsmith@example.com
Subject: Email is cool
This is an email message.
.
250 SAA112345 Message accepted for delivery
```

服务器回应了。它请求输入邮件正文，然后在—行中标记一个单独的句点以表示邮件的结尾。

可以看到我们传送了一些清晰的缺省报头，如日期、To 和 From 报头、主题和邮件正文。最后指定了一个句点。服务器回应说它已经接受了该邮件。然后每个电子邮件被提交给一个邮件队列，被处理，最后由服务器转送。

现在你可以重复 MAIL FROM 命令继续发送电子邮件，或者使用 QUIT 命令从服务器断开。

```
QUIT
221 Goodbye
```

这是一个简单的发送电子邮件的情景。它是可能发生的最基本的命令交换。当把像加密和身份验证之类的元素考虑进来时，大多数情况下常规的电子邮件发送过程都会更复杂一些。

补充知识：电子邮件地址

那么作为一个电子邮件地址怎样算是可以接受呢？什么是一个格式正确的电子邮件地址？一个电子邮件地址最基本的格式是一个用户名和一个主机名、域名或者完整合格的域名，并以@符号隔开，如 jsmith@example.com。允许什么样的字符与接受什么样的邮件地址格式这两条规则常常混淆。电子邮件地址可以采取许多格式，不同的电子邮件服务器或 MTA 接受不同的格式作为有效格式。将要介绍的 MTA Postfix 接受很多种格式的邮件地址。Postfix 的邮件地址重写指南演示了其中一些可接受的电子邮件地址格式：http://www.postfix.org/ADDRESS_REWRITING_README.html。

10.1.2 电子邮件发送之后发生了什么事？

服务器接受来自客户端的电子邮件并把它放到邮件队列之后，就开始执行一整套新命令和步骤。首先，服务器要找到邮件的发送地点。为此，服务器拿出电子邮件地址中@符号右边那部分。它一般是完整合格的域名，如 example.com。然后，电子邮件服务器利用 DNS 查询联系远程域并询问应该把邮件发送到哪儿。

■注：第 9 章介绍过 DNS。

为此，电子邮件服务器查询一个特别的 DNS 记录，即 MX 记录。查询结果会返回一个或更多输入项，以告诉电子邮件服务器邮件发往的位置，这通常是一个特定的主机或 IP 地址。如果返回电子邮件服务器超过 1 个，那么同时会返回一个优先级，以告诉你的电子邮件服务器先使用哪个输入项，然后使用哪个，依此类推。

■注：如果 DNS 查询结果表示没有 MX 记录，电子邮件服务器就不会投递这封邮件，并发送一个消息通知你。这通常在用户打字错误或指定了不正确的邮件地址的情况下发生。

然后电子邮件服务器把往外走的电子邮件提交给别的队列，从那里它被转送给目的邮件服务器。为此，邮件服务器尝试通过 TCP 端口 25 按照 MX 记录指定的优先级顺序与 MX 查询所返回的每个邮件服务器连接。然后邮件服务器密切注意一个提交队列，看看是否投递电子邮件。

1. 如果某个电子邮件服务器回应了，它就尝试提交邮件。

- 2. 如果某个邮件服务器没有回应，它会按顺序尝试 MX 记录所返回的下一个服务器。
- 3. 如果没有邮件服务器回应，它通常会把邮件放到队列里，稍后再试。
- 4. 如果经过连续的失败这封邮件仍然不能投递，它会通过一封电子邮件向用户报告失败的消息。

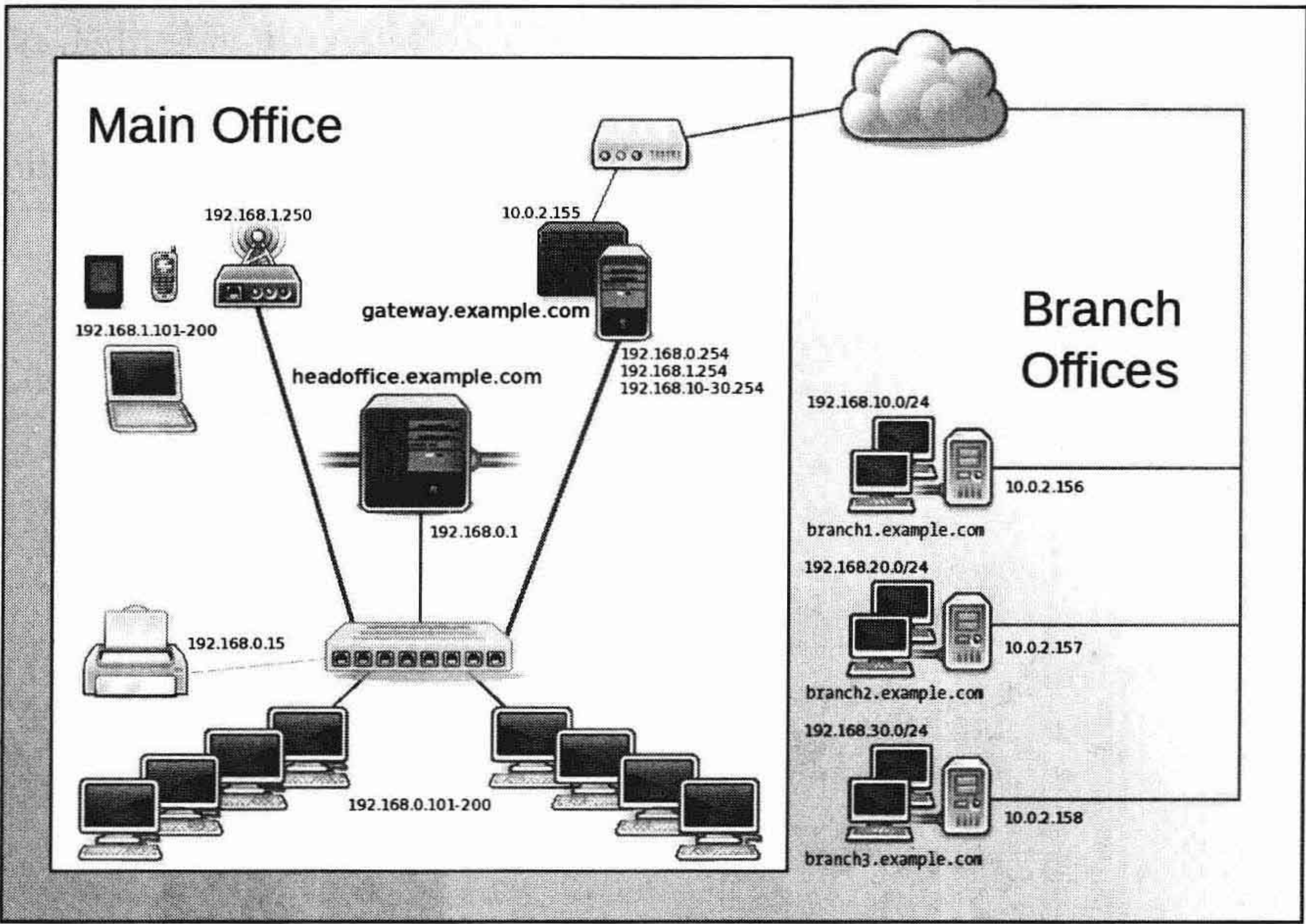
电子邮件服务器尝试发往的目的服务器可能是邮件最终的目的地。或者，它可能只是一个网关，通过它邮件被转发给 1 个或更多邮件服务器直到最终到达它的目的地。这取决于目的地如何配置它的邮件平台。许多平台都有一个面向 Internet 的电子邮件网关，它接收邮件，然后有一个内部邮件服务器处理内部邮件。这样的配置允许在电子邮件网关上添加一些功能，如垃圾邮件或病毒过滤，这与内部服务器上可能拥有的功能不同。

10.2 配置电子邮件

下面我们将介绍如何创建一个基本的电子邮件服务器配置，使用户可以发送和接收电子邮件，并帮助保护用户免受垃圾邮件、病毒和恶意软件的侵害。我们还将使用传输层安全 (Transport Layer Security, TLS) 协议 (一种用来加密邮件的加密形式)，并讨论 SMTP AUTH (一种在用户发送电子邮件时对它们进行身份验证的方式)。

我们将使用 headoffice.example.com 主机作为电子邮件服务器，gateway.example.com 会把电子邮件流量传送给它，这在第 6 章配置范例网络时介绍过。

邮件服务器被称作 mail.example.com (它是在第 9 章创建的一个 DNS CNAME)，IP 地址是 192.168.0.1。从图 10-2 可以看到这个范例网络。



Main Office: 主办公室; Branch Offices: 分支办公室
图 10-2 范例网络

10.2.1 安装

我们将从安装所需的 Postfix 和 Dovecot 应用程序开始。

1. Red Hat 安装

在 Red Hat Enterprise Linux 上，默认的 MTA 是 Sendmail，但我们要安装 Postfix 并修改默认的 MTA。这样做的理由是，根据我们的经验，Postfix 更易理解、配置和检修。

■注：关于哪个 MTA 最好有很多争论，这涉及许多 MTA，包括 Sendmail、Postfix、Exim 以及其他 MTA。我们推荐从 Postfix 开始，然后随机应变。你会发现 Postfix 容易用并能满足所有的需求。

首先，让我们看看 Postfix 和 Dovecot 是否已经安装。

```
$ sudo rpm -q postfix dovecot
postfix-2.3.3.2
package dovecot is not installed
```

如果 1 个或更多软件包还没有安装，就要安装它们。在这里我们使用 yum 命令来完成。

```
$ sudo yum install postfix dovecot
```

■注：另外一些作为先决条件的软件包也可以通过这些命令安装。本章稍后还会安装一些其他软件包以支持其他功能。

现在要告诉 Red Hat，Postfix 是默认 MTA。我们使用一个叫作 Mail Transport Agent Switcher 或 system-switch-mail 的命令做这件事。它可以通过 GUI 或使用叫 system-switch-mail-nox 的命令在命令行里启动。

这个转换器可以从 GUI 启动。通过菜单命令选择“System”→“Administration”→“Mail Transport Agent Switcher”，如图 10-3 所示。



图 10-3 启动 Mail Transport Agent Switcher

从图 10-4 可以看到转换器的对话框。

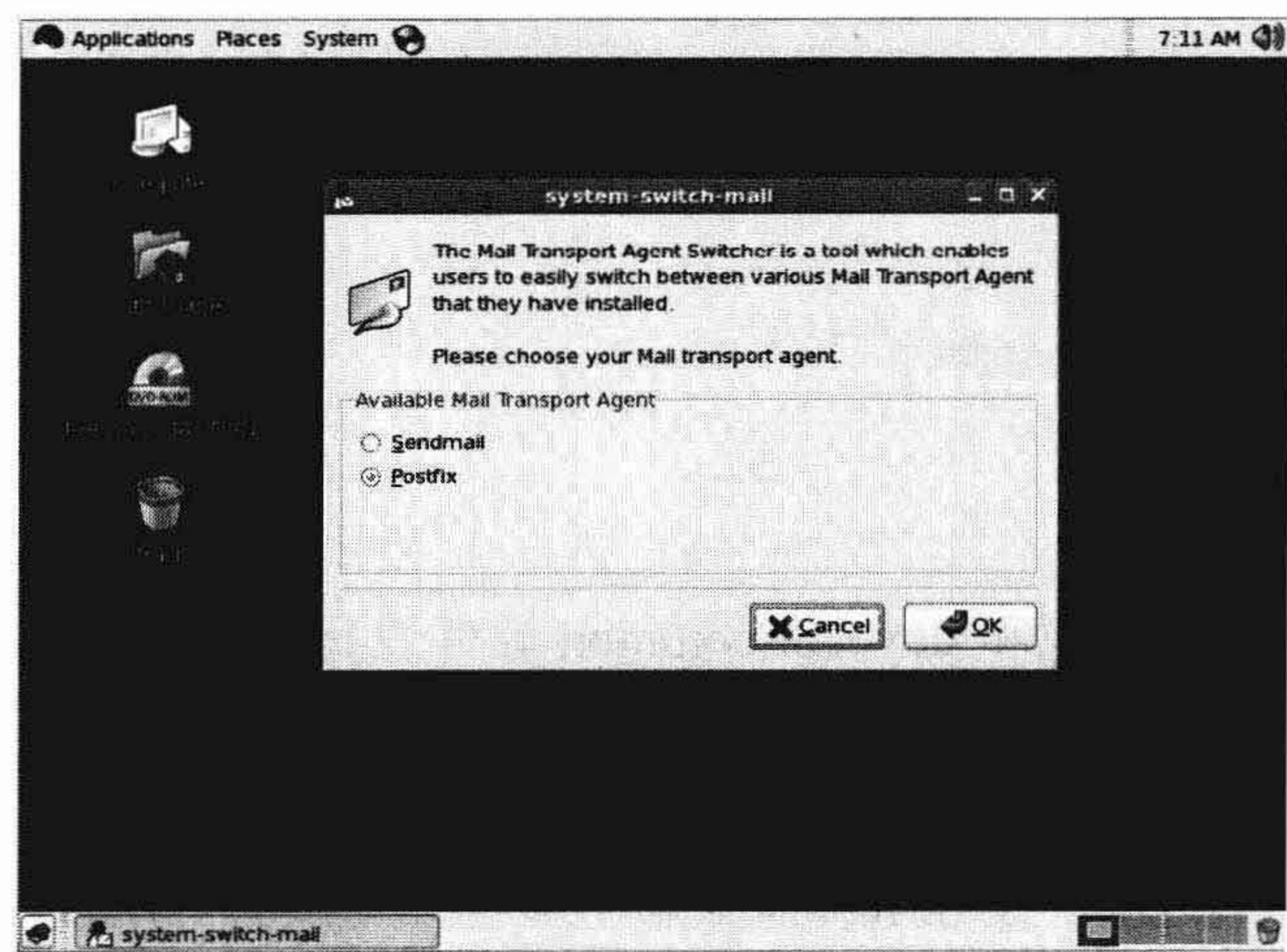


图 10-4 Mail Transport Agent Switcher 对话框

我们在对话框中选择“Postfix MTA”并单击“Ok”。这就把默认 MTA 改为了 Postfix，替代了已有的 Sendmail MTA。

或者你可以在命令行中运行 system-switch-mail-nox 命令。

```
$ sudo system-switch-mail-nox
```

从图 10-5 可以看到命令行对话框。

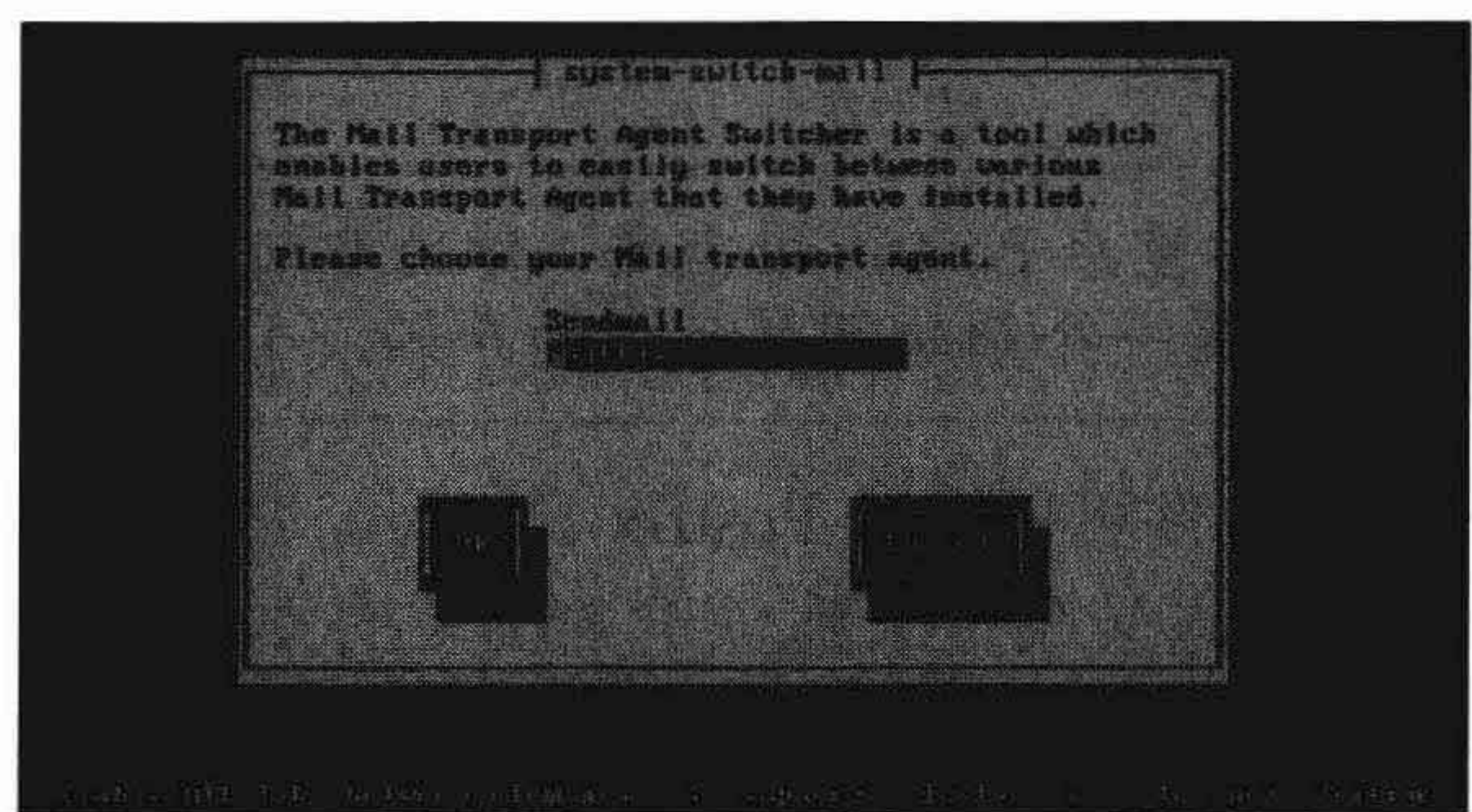


图 10-5 system-switch-mail-nox 对话框

提示：如果主机不能找到这些目录，就要分别为命令行和 GUI 程序安装 system-switch-mail 和/或 system-switch-mail-gnome 软件包。

2. Ubuntu 安装

Postfix MTA 是 Ubuntu 发行版上默认的 MTA，因此通常已经安装好了。的确，在第 2 章 Ubuntu 安装过程中得到过执行一些基本 MTA 配置的提示。我们很快就会重访那个配置会话（见补充知识“编辑 Postfix 配置”）。

如果没有安装的话，那么我们希望安装 Postfix 和 Dovecot 软件包。首先检查一下，看看它们是否已经安装。

```
$ sudo aptitude show postfix
Package: postfix
State: installed
...
$ sudo aptitude show dovecot-common
Package: dovecot-common
State: installed
...
```

在这里检查 postfix 软件包和 dovecot-common 软件包是否已安装。如果两者都安装了，那么你就已经得到所需的东西。

如果没有安装，你可以使用 aptitude 命令安装它们。

```
$ sudo aptitude install postfix dovecot-common dovecot-imapd dovecot-pop3d
```

10.2.2 启动 Postfix

安装之后，我们使用第 5 章介绍的服务管理工具很容易开启 Postfix。对于 Red Hat，我们使用 service 命令启动 Postfix。

```
$ sudo service postfix start
```

对于 Ubuntu，我们使用 invoke-rc.d 命令。

```
$ sudo invoke-rc.d postfix start
```

提示：在对配置做完修改之后重启 Postfix 很重要，你会看到这个提示贯穿全章。如果不能找到出故障的原因，重启通常是一个解决问题的好的开始。

我们通过检查它的日志文件可以确认 Postfix 已启动。Postfix 把它的输出记录到系统记录器（或系统日志），而日志又被记录到位于主机上的文件中。你可以在 /var/log 目录里找到它的日志文件。在 Red Hat 上，我们需要查看 /var/log/maillog 文件。在 Ubuntu 上，Postfix 把所有日志消息发送到 /var/log/mail.log。在 Ubuntu 上，错误和警告消息也被分别记录到 /var/log/mail.err 和 /var/log/mail.warn 文件。监视这些文件中的错误消息是一个好主意。而一个很好的执行方式就是使用带 -f 选项的 tail 命令，它实时监视文件，并在新的日志消息添加到文件时往下翻页。

```
$ sudo tail -f /var/log/maillog
```

如果 Postfix 成功开启，你应该看到下面的日志消息。

```
Jan 5 19:52:33 au-mel-rhel-1 postfix/postfix-script[31669]: starting the Postfix mail system
Jan 5 19:52:33 au-mel-rhel-1 postfix/master[31670]: daemon started -- version 2.5.5, configuration /etc/postfix
```


10.2.3 了解 Postfix 配置

大部分 Postfix 配置都由位于/etc/postfix 目录里的两个文件处理。这两个文件叫作 main.cf 和 master.cf。main.cf 文件包含可用来定制 Postfix 的主要配置选项的一个子集。你可以往这个文件添加配置 Postfix 所需的任何附加选项。master.cf 文件管理客户端如何连接到服务器以及如何配置构成服务器的服务。大部分时间你都不会怎么修改 master.cf 文件。

让我们先快速看一看每个文件的构成。

■注：大部分发行版都安装了已配置过的 main.cf 和 master.cf 文件。它们通常包含全面描述每个选项的内置文档。

在 main.cf 中，每个选项都按照如下格式构造。

```
option = value
```

选项不必按顺序放置，并且空行会被忽略。

■注：与大部分 Linux 配置文件一样，任何以#开头的行都是注释。

从列表 10-2 可以看到一个典型的 main.cf 文件实例。

列表 10-2 main.cf 文件

```
# The command_directory parameter specifies the location of all
# postXXX commands.
#
command_directory = /usr/sbin
```

本章中将涉及 main.cf 文件里的一些配置选项；完整的选项清单可以在 <http://www.postfix.org/postconf.5.html> 找到。

在 master.cf 文件中可以看到一个能在 Postfix 里打开和配置的后台程序、服务和进程的结构化列表。列表 10-3 给出了源自该文件的一些样例。

列表 10-3 master.cf 文件

```
# service type private unpriv chroot wakeup maxproc command + args
smtp      inet  n       -       n       -       -       smtpd
submission inet  n       -       n       -       -       smtpd
-o smtpd_enforce_tls=yes
-o smtpd_sasl_auth_enable=yes
```

这里定义了两个服务，smtp 和 submission。一般每个服务都是一个运行在后台的执行特定功能的后台进程。

■注：与其他许多服务不一样，Postfix 不是单个独立的后台程序，而是一个执行单独功能并互相通信的小进程的集合。因此，在打开 Postfix 时你可以看到它启动了许多进程而不是单个进程。

在本例中，smtp 服务提供基本的 SMTP 服务，为主机接收 TCP 端口 25 上的邮件，而

submission 服务是一个备选服务，它在 TCP 端口 567 上监听，有时用来接收来自内部邮件客户端的电子邮件。

这一行接下来定义了服务的类型；这里的 inet 把它们定义为基于网络的服务（即运行在网络接口上的服务）。同样可用的其他类型的服务包括本地 Unix 套接字和有名管道（见 http://en.wikipedia.org/wiki/Unix_domain_socket 和 http://en.wikipedia.org/wiki/Named_pipe）。

接下来我们对每个服务进行了一连串的设置。每个设置在一行里按顺序指定，用空格分开。你不用担心这些基本配置的设置，在 <http://www.postfix.org/master.5.html> 上可以找到更多有关它们的资料。

最后，每个服务需要指定启动该服务的命令以及传递给该命令的参数。在列表 10-3 中，这两个服务都是用命令 smtpd 开启的，而 submission 服务有几个可传递的选项。这些选项和 main.cf 配置文件里的可用选项相同，这为配置特定的服务提供了一种不同于主配置的方式。本章稍后的“防止病毒和垃圾邮件”一节会讲述如何把这些选项传递给个别服务。

补充知识：编辑 Postfix 的配置

有 3 种方式可以编辑 Postfix 配置文件，特别是 main.cf 文件。

- 可以使用 postconf 命令。
- 在 Ubuntu 上，dpkg-reconfigure 命令可以执行基本的 Postfix 配置。
- 可以使用如 vim 或 gedit 之类的文本编辑器。

我们一般推荐最后这种方式，使用一个文本编辑器编辑配置文件，但首先让我们快速看一下另外两个选项。

postconf 命令是允许对 main.cf 操作以及显示已有配置的特别命令。该命令还有一些有用的命令行选项，可用来调试和显示配置。例如，为了显示从默认配置更改过的配置选项（即为主机所做的改动），我们可以使用 -n 配置标志。

```
$ sudo postconf -n
```

如果正在查找故障，通常要求提供 postconf -n 命令的输出。你还可以使用 -d 标志显示每个配置选项和它的默认设置。

最后，你可以使用该命令与 -e 选项真正编辑配置文件，如下所示。

```
$ sudo postconf -e 'inet_interfaces = all'
```

在这里 inet_interfaces 选项（它决定 Postfix 把自身连接到哪个网络接口上）的值被设置为 all。

在 Ubuntu 上，你可以使用 dpkg-reconfigure 命令配置 Postfix 的基本状态。

```
$ sudo dpkg-reconfigure postfix
```

使用此命令可以选择一个基本配置模型，如收发电子邮件的基本 Internet 主机。另一个模型使用 smarthost；在这种情况下，电子邮件被发送给一个中间的中继服务器。这个模式经常用在 ISP 环境中，在那种环境下有时限制直接从端口 25 收发邮件。这样的设计阻止 DSL、ADSL 或有线网络上受损害主机发送和转发电子邮件，从而减少垃圾邮件的数量。本章稍后的补充知识“Smarthosting”将介绍 smarthost。

其他许多配置选项也可以由同样的工具设置。不过我们不推荐使用这个工具；相反，应使用文本编辑器手动编辑配置。

永远要记住，修改完 Postfix 配置必须重启 postfix 服务。

10.2.4 初始配置

尽管看起来配置过程和配置选项都很复杂，其实 Postfix 非常容易配置。为一个域设计一个收发邮件的简单电子邮件服务器在数分钟内就可以设置好。

第一步是告诉 Postfix 应该为什么域处理邮件。为此，我们通过添加域的方式更新 `main.cf` 配置文件中的 `mydestination` 配置选项，本例中的域是 `example.com`。我们可以通过直接编辑 `main.cf` 文件修改该选项。

```
mydestination = mail.example.com, localhost.localdomain, localhost, example.com
```

我们还可以使用 `postconf` 命令修改它。

```
$ sudo postconf -e "mydestination = mail.example.com, localhost.localdomain, localhost, example.com"
```

上一行添加了两项：邮件服务器和本地域。Postfix 现在知道它是否接受目的地为这些地址的邮件，例如对于发给 `ataylor@mail.example.com` 或 `ataylor@example.com` 的邮件，它应该接受并处理它。

注：Postfix 只接收它所知道的用户的邮件。一般情况下，它必须是在该主机上创建的用户，但是本章后面的“虚拟域和用户”一节将简要地介绍“虚拟”用户。在某些情况下，Postfix 知道其他电子邮件服务器上的用户，可以被配置为转发发往其他主机的邮件。但是一般情况下，如果 Postfix 不能找到邮件的收件人（例如，如果一封邮件是发送给 `bjones@example.com` 的，而 Postfix 不知道这个用户），这封邮件就会被丢弃。

如果希望 Postfix 接收其他域的电子邮件，比如说，希望接收 `example.net` 域的邮件，我们可以把那个域添加到 `mydestination` 选项中。

```
mydestination = mail.example.com, localhost.localdomain, localhost, example.com, example.net
```

通常已经存在的其他输入项包括 `localhost`，它告诉 Postfix 处理主机本地的电子邮件，如本地进程发送的电子邮件。

接下来我们需要把本地网络添加到 `mynetworks` 配置选项。它告诉 Postfix 处理哪些 IP 地址范围的邮件。在我们的例子中，我们只关心 `192.168.0.0/24` 和 `192.168.1.0/24` 这两个范围，二者是本地的有线和无线网络段。`127.0.0.1` 或 `localhost` 地址也应该在这个选项里。

```
mynetworks = 127.0.0.0/8, 192.168.0.0/24, 192.168.1.0/24
```

或者我们可以再次使用 `postconf` 命令编辑此选项。

```
$ sudo postconf -e "mynetworks = 127.0.0.0/8, 192.168.0.0/24, 192.168.1.0/24"
```

现在我们需要把 Postfix 连接到希望监听的网络接口。本例打算连接所有的网络接口。

```
inet_interfaces = all
```

或者使用 `postconf` 命令：


```
$ sudo postconf -e "inet_interfaces = all"
```

不过我们可以有更多的选择。如果只想把 Postfix 连接到单个接口上，我们可以照下面这样做。

```
inet_interfaces = eth0
```

■提示：如果正在使用 IPv6，通过把 `inet_protocols` 选项改为 `all` 可以打开它。默认情况下只打开 IPv4。

最后，做好了所有相关的改动之后，我们要重启 Postfix。在 Ubuntu 上，我们可以发出下面的命令。

```
$ sudo invoke-rc.d postfix restart
```

在 Red Hat 衍生的发行版上，我们发出下面的命令。

```
$ sudo service postfix reload
Reloading postfix: [ OK ]
```

■注：每次修改 Postfix 的配置选项后，我们都应该重启 Postfix。

除了重启 Postfix 之外，我们还需要确保主机上的 TCP 端口 25 是打开的，以允许进来的连接。如果 iptables 防火墙在运行，我们就需要创建允许访问的规则，如下所示。

```
$ sudo iptables -A INPUT -p tcp -m state --state NEW -m tcp --dport 25 -j ACCEPT
```

■提示：第 6 章介绍过 iptables 防火墙、规则的制定和端口的开启。

10.2.5 测试 Postfix

现在我们可以通过给自己发送一封电子邮件测试一下，看看 Postfix 是否在工作。我们将使用一个叫 netcat 或简称为 nc 的工具来做这件事。nc 命令是网络工具里的“瑞士军刀”，可以用来处理 TCP 和 UDP 连接。

■注：还有其他发送测试邮件的方式（例如，在命令行里使用 mail 命令），不过 nc 命令允许查看 SMTP 命令，并在命令行上显示所有的错误消息。

我们可以使用 nc 命令创建一个与 Postfix 电子邮件服务器的会话，如列表 10-4 所示。

列表 10-4 使用 nc 创建的电子邮件会话

```
$ nc mail.example.com 25
220 mail.example.org ESMTP Postfix
ehlo example.com
250-mail.example.com
250-PIPELINING
```



```

250-SIZE 10240000
250-VRFY
250-ETRN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
mail from: jsmith@example.com
250 2.1.0 Ok
rcpt to: ataylor@example.com
250 2.1.5 Ok
data
354 End data with <CR><LF>.<CR><LF>
Subject: My first mail for my domain
This is a test.
Thanks
Mr Testing
.
250 2.0.0 Ok: queued as EEC04A0BA0
quit
221 2.0.0 Bye

```

在列表 10-4 中，我们先指定 `nc` 命令、希望连接到的电子邮件服务器和端口 25。接下来，按照发送一封电子邮件所需的 SMTP 命令逐步调试。在列表 10-4 中凡是加粗的文本都表示用户输入的命令，因为发送电子邮件需要输入这些文本。你要根据自己的情况调整这些文本，例如把 `example.com` 替换为你的域名。没加粗的文本是期望从电子邮件服务器得到的回应。

■注：你可以看到每一行的前缀都是 SMTP 回应码，如 250。这些代码的完整列表可以在 <http://www.unixhub.com/docs/email/SMTPcodes.html> 上看到。

用 EHLO 命令告诉服务器我们是谁，它用它自己的身份和可用功能清单作为回应。然后我们表明电子邮件的收件人和发件人。在本例中，邮件来自 `jsmith@example.com`，要发送给 `ataylor@example.com`。

■注：这些用户是在第 4 章创建的。如果一直在遵照本书操作，你应该替换为本地主机上的用户名或者创建一些新用户来测试电子邮件。

然后我们用 DATA 命令告诉服务器将要发送这封电子邮件。输入邮件内容，用一个句点标记它的结尾，然后退出。

■提示：如前所述，另一种从命令行发送本地电子邮件的方式是使用 `mail` 命令。它没有提供一个诊断服务器的视角，但是用它发送邮件比较容易。要启动这个命令，在命令行里键入 `mail` 与想要发往的地址，如 `mail root@example.com`。只输入 `mail` 命令本身将打开一个非常简单的命令行邮件客户端。

现在我们可以检查邮件是否已经达到。在本例中，我们希望检查邮件是否已经被用户 `ataylor` 接收。默认情况下，Red Hat 和 Ubuntu 两者都使用一个被称为 `mbox` 的邮箱格式存储接收到的电子邮件。`mbox` 格式把所有的电子邮件保管在单个文件中。但令人混淆的是，虽然

每个发行版都把电子邮件存储在 mbox 文件里，但这些文件的位置因发行版的不同而不同。

■注：关于 mbox 格式的更多资料请参阅 <http://en.wikipedia.org/wiki/Mbox>。

在 Ubuntu 上，以用户名命名的邮箱文件被创建在 /var/mail 目录，如 /var/mail/ataylor。

在 Red Hat 上，每个用户都有一个 mbox 格式的邮箱文件包含在 /var/spool/mail 目录中。此文件以用户名命名，如 /var/spool/mail/ataylor。

■注：在 Ubuntu 上，/var/spool/mail 还是 /var/mail 的一个符号链接。

让我们通过列表 10-5 看一看其中一个 mbox 文件的内容。

列表 10-5 一个 mbox 文件

```
$ more /var/mail/ataylor
From jsmith@example.com Mon Aug 11 05:02:26 2008
Return-Path: <jsmith@example.com>
Received: from mail.example.com (mail.example.com [127.0.0.1])
by mail.example.com (8.13.8/8.13.8) with ESMTP id m7AJ2QK020858
for <ataylor@example.com>; Mon, 11 Aug 2008 05:02:26 +1000
Received: (from jsmith@example.com)
by mail.example.com (8.13.8/8.13.8/Submit) id m7AJ2Q2V020857
for ataylor; Mon, 11 Aug 2008 05:02:26 +1000
Date: Mon, 11 Aug 2008 05:02:26 +1000
From: jsmith <jsmith@example.com>
Message-Id: <200808101902.m7AJ2Q2V020857@example.com>
To: ataylor@example.com
Subject: My first mail for my domain
Status: O

This is a test.
Thanks
Mr Testing
```

从列表 10-5 可以看到，mbox 文件包含了我们的电子邮件，包括邮件的所有报头和正文。其他电子邮件会追加到该文件中。这样做的后果之一就是，如果有大量的电子邮件，这些文件就会变得很难管理、搜索和备份。

补充知识：别名

当电子邮件到达主机时，Postfix 查看收件人以决定把它投递给谁。一些程序使用像邮件管理员地址（例如，MTA 的错误通常被发送给域上的用户邮件管理员）和 root 用户这样的收件人。因为人们一般不会以这些用户的身份登入，所以这封电子邮件就可能永远不会被看到。Postfix 使用一个叫作别名的功能允许把发送给这些收件人的邮件重定向到其他用户。Postfix 使用 main.cf 配置文件中一个叫作 alias_maps 的配置选项指定一个文件，来匹配收件人与本应接收这封邮件的真正用户。默认的别名文件通常是 /etc/aliases。在这个文件中你可以找到一个用户列表，如下所示。

```
user1:user2
```

在这里每个用户名都用冒号分开，所有给 user1 的邮件都会重定向给 user2。你应该检查一下这个文件，以确保邮件管理员和 root 用户的电子邮件被重定向到合适的用户。你还可以通过

指定一个外部电子邮件地址（如 `username@example.com`），给其他主机上的用户发送电子邮件。通常的方式是把系统用户所有的电子邮件都导向 `root` 用户，然后把 `root` 用户重定向到需要看到这封邮件的用户，如下所示。

```
postmaster: root
operator: root
lp: root
root: ataylor
```

在这里 `postmaster`、`operator` 和 `lp` 用户的电子邮件都重定向到 `root` 用户，而 `root` 用户的邮件又重定向到用户 `ataylor`。

对 `/etc/aliases` 文件做了改动之后，你要运行一个叫作 `newaliases` 的命令，以便使用这些改动更新 Postfix。

```
$ sudo newaliases
```

10.2.6 选择邮箱格式

除了潜在的变成巨大而难处理文件的问题，默认的 `mbox` 格式还有另外一个问题：文件损坏的可能性。例如，如果 MTA 向 `mbox` 文件投递消息的同时，MUA 或邮件客户端正在删除消息，那么 `mbox` 文件就有可能损坏或返回不可预期的结果。

一个可替代 `mbox` 的方案是 Maildir 邮箱格式。Maildir 在一个目录下的多个独立的文件中保存电子邮件，而不是使用单个文件。这样就允许多个进程与邮箱交互而没有冲突和损坏的风险。它也更容易备份和恢复。

Maildir 格式是一个恰好叫作 Maildir 的目录，它包括 3 个子目录 `cur`、`new` 和 `tmp`。这里显示了 Maildir 格式的目录列表。

```
$ ls -l Maildir
total 168
drwxr-xr-x 2 ataylor ataylor 28672 2009-01-01 13:53 cur
drwxr-xr-x 2 ataylor ataylor 4096 2009-01-01 13:53 new
drwxr-xr-x 2 ataylor ataylor 4096 2009-01-01 13:53 tmp
```

电子邮件消息首先被投递到 `tmp` 目录并起一个唯一的名字（通常根据当前时间、主机名和其他伪随机特征构造）。然后邮件被转移到 `new` 目录，在那里它们处于“未读”状态。当 MUA 或邮件客户端连接到这个邮箱时，它检测位于 `new` 目录中的邮件，然后把它移动到 `cur` 目录。

注：这听起来有点复杂，但是它确保了邮件在投递、阅读、发送和从邮箱删除时不会被损坏或者放错地方。

为了使用 Maildir 而不是 `mbox`，要告诉 Postfix 我们在使用一种不同的邮箱格式。我们还要把默认的邮箱位置从它现在的位置（`/var/mail` 或 `/var/spool/mail`）改到用户的主目录。为了做这两件事，我们要更新一个叫作 `home_mailbox` 的 Postfix 选项，如下所示。

```
home_mailbox = Maildir/
```


或者使用 `postconf` 命令：

```
$ sudo postconf -e "home_mailbox = Maildir/"
```

`home_mailbox` 选项告诉 Postfix 用户邮箱相对用户主目录的位置，从而 Maildir/被翻译为 /home/ataylor/Maildir。

■注：结尾的/很重要。它必须在那里，以便告诉 Postfix 该目录是一个 Maildir 目录。

我们还需要确认另一个选项 `mailbox_command` 是空白的。`mailbox_command` 选项可以指定一个外部命令，如 `procmail` 或 `maildrop` 之类的邮件处理工具。这些被称为 MDA 或邮件过滤器的工具可以在把邮件投递到邮箱时对它进行相应处理。本章稍后的补充知识“邮件投递代理和邮件过滤”将更详细地讲述这些程序。那么现在把这个选项设置为空值。

```
mailbox_command =
```

或者我们还使用 `postconf` 命令。

```
sudo postconf -e "mailbox_command = "
```

最后，我们要重启 Postfix 以使所有这些生效，比如在 Red Hat 上，命令如下所示。

```
$ sudo service postfix restart
```

现在，如果再发送一封电子邮件，你可以发现在收件人的主目录里新创建了一个叫作 Maildir 的目录。这个目录里面是 `tmp`、`new` 和 `cur` 子目录，`new` 目录里是这封邮件，它是一个单独的文件。你可以使用 `less` 命令显示这个文件的内容，如下所示。

```
$ less /home/jsmith/Maildir/new/1231486748.M906603P5576.mail.example.com,W=8271:2,Se
From jsmith@example.com Wed Aug 14 03:08:12 2008
Return-Path: <jsmith@example.com>
Received: from mail.example.com (mail.example.com [127.0.0.1])
by mail.example.com (8.13.8/8.13.8) with ESMTTP id m7AJ2QQK020858
for <ataylor@example.com>; Wed, 13 Aug 2008 03:08:12 +1000
Received: (from jsmith@example.com)
by mail.example.com (8.13.8/8.13.8/Submit) id m7AJ2Q2V020857
for ataylor; Wed, 13 Aug 2008 03:08:12 +1000
Date: Mon, 11 Aug 2008 05:02:26 +1000
From: jsmith <jsmith@example.com>
Message-Id: <200808101902.m7AJ2Q2V0304567@example.com>
To: ataylor@example.com
Subject: My first email to my new Maildir mailbox
Status: 0

This is also a test.
Thanks
Mr Testing
```

在这里可以看到，个人的 Maildir 文件包含一封携带了全部报头的电子邮件。

■注：每个 Maildir 必须由它所属的用户所有。例如，属于 ataylor 用户的 Maildir 目录必须只为 ataylor 所有，并只对 ataylor 可写（有 0700 的权限）。如果需要的话，你可以使用 `chown` 和 `chmod` 命令分别修改所有权和权限。

尽管在本例中 Postfix 自动创建了 Maildir 目录，但是预先用一个空的 Maildir 目录构造用户主目录通常是个好主意。你可以通过往/etc/skel 目录添加一个空的 Maildir 目录来做到这一点。我们在第 4 章看到过/etc/skel 目录，知道它的内容被复制到新创建用户的主目录。

为了在 Ubuntu 上创建新的 Maildir 目录，有一个叫作 maildirmake.dovecot 的命令可以自动创建 Maildir 的结构。

```
$ sudo maildirmake.dovecot /etc/skel/Maildir
```

这个命令还创建 tmp、new 和 cur 子目录。

提示：你也可以在 Maildir 结构里创建文件夹，如一个 Sent 文件夹。文件夹可用于电子邮件的排序和存储以便更容易查找电子邮件。文件夹是前缀为一个句点(.)的子目录；使用命令 maildirmake.dovecot /etc/skel/Maildir/.Sent 所创建的 Sent 文件夹的位置是/home/ataylor/Maildir/.Sent。要小心避免文件夹名包含空格，如“My Personal Mail”，因为有时邮件客户端会弄不清楚。你应该使用下划线或短划线把单词连接在一起，就像这样“My-Personal_Mail”。

在 Ubuntu 上，有另外一种可能更容易的方式创建 Maildir 目录和文件夹。为此，你可以安装一个叫 maildrop 的软件包。它包含 maildirmake.dovecot 命令的高级版本：maildirmake 命令。

在 Red Hat 上，同样的命令也叫 maildirmake。如果安装了 maildrop 软件包，我们就可以使用它。这个软件包在 Red Hat 不能直接用，但可以从第三方 RPM 软件资料库下载一个兼容的软件包。

```
$ sudo rpm -Uvh ftp://ftp.silfreed.net/repo/rhel/5/i386/silfreednet/RPMS/
maildrop-2.0.4-1.el5.i386.rpm
```

这里这个软件包适用于 Red Hat Enterprise Linux 5（见 RPM 名字中的 el5）。我们使用 rpm 命令下载并安装它。

注：这是一个第三方 RPM 软件资料库，而 Red Hat 不支持 RPM。如果对此感觉不舒服，你还可以从 <http://www.courier-mta.org/download.php#maildrop> 下载 Courier Maildrop 源代码，使用 <http://www.courier-mta.org/rpm.html> 中的说明以及第 7 章介绍的方法创建自己的 RPM。

然后你可以使用 maildirmake 命令创建目录框架和必需的文件夹。

```
$ sudo maildirmake /etc/skel/Maildir
$ sudo maildirmake -f Sent /etc/skel/Maildir
$ sudo maildirmake -f Trash /etc/skel/Maildir
$ sudo maildirmake -f Drafts /etc/skel/Maildir
$ sudo maildirmake -f Spam /etc/skel/Maildir
```

这里使用 maildirmake 的 -f 选项创建文件夹。我们指定了想要创建的文件夹名与它所在的 Maildir。

如果不想安装 maildrop 软件包，你可以使用 mkdir 命令创建这些目录，如列表 10-6 所示。

列表 10-6 手动创建 Maildir

```
$ sudo mkdir -p /etc/skel/Maildir/{cur,new,tmp}; chmod -R 0700 /etc/skel/Maildir
```

■注：列表 10-6 中使用了一个灵巧的 Bash 快捷方式，用大括号（{ }）把以逗号隔开的全部 3 个目录列了出来。这个大括号扩展技巧告诉 mkdir 创建全部 3 个子目录。此技巧可以在其他各种命令中使用，而无需输入 3 次命令。-p 选项创建全部父目录。

列表 10-6 创建了 Maildir 和必需的子目录，并且把获得的目录权限改为 0700，这样就只允许拥有这个 Maildir 的用户可以访问它。

补充知识：邮件投递代理和邮件过滤

本章早前提到了 MDA。这个工具位于 MTA 和用户邮箱之间，你可以用早先看到的 mailbox_command 配置选项告诉 Postfix 使用什么样的 MDA，如果有的话。如果在这个选项中指定了 MDA，Postfix 就会给 MDA 投递邮件，而 MDA 会把邮件投递给用户的邮箱。

在投递过程中，MDA 可以执行多种动作；例如，它可以寻找电子邮件的特征（比如它来自谁），然后把它重定向到一个具体的邮箱文件夹。许多人使用 MDA 把邮寄列表里的电子邮件分类放到独立的文件夹里。MDA 还用来根据其他程序添加的报头为电子邮件分类；例如，许多垃圾邮件过滤器会添加表示邮件是否为垃圾邮件的报头。MDA 可以阅读这些报头，并把电子邮件放到一个合适的文件夹里——例如，放到一个垃圾邮件文件夹里。使用 MDA 还可以生成邮局通知，给其他人转发特别的邮件，以及执行其他各种任务。

某些 MDA 还可以调用其他程序。例如，有些人不在他们的 MTA 上运行垃圾邮件过滤器，而是用他们的 MDA 运行它。

有两种流行的 MDA：procmail 和 maildrop。二者在 Red Hat 和 Ubuntu 上都可用。本章稍后的“使用 procmail 和 maildrop”一节将说明如何配置和使用 procmail 和 maildrop，以帮助用户处理垃圾邮件。

10.3 Postfix 扩展配置

迄今为止，我们简要介绍了 Postfix 的基本配置，能让本地用户收发电子邮件。然而，Postfix 有很多其他功能，可以使环境更安全，使用户体验更好，包括如下内容。

- 加密：保护电子邮件和用户机密资料的传输。
- 身份验证：确保只有合适的和经过验证的用户可以发送电子邮件。

10.3.1 使用加密功能

Postfix 的 MTA 可以通过一个叫作 TLS 的加密协议为电子邮件的传输加密。TLS 是 SSL 的后继协议，通常用于加密 TCP/IP 流量。以前在使用安全套接字层上的超文本传输协议（Hypertext Transfer Protocol over Secure Socket Layer, HTTPS）连接一个网站（如 https://www.gmail.com）时，已经用到了 TLS，尽管你可能没有意识到。许多老的 HTTPS 连接还在使用

老的 SSL 协议，但是有些连接下面运行的是新的 TLS。

TLS 为电子邮件通信提供了两个关键的功能。

- 防止窃听电子邮件的内容。
- 为客户端和服务端之间的通信加密，从而保护身份验证的过程。

1. TLS 和证书

与 SSL 一样，TLS 使用数字证书和一种叫作公钥加密的密码系统。公钥加密有两个密钥：公开密钥可公用，私有密钥存储在服务器上并保密。用公钥加密的内容只能用相对应的私钥解密。

■注：数字证书和公钥密码是个复杂的话题。这里只能简要介绍 TLS 的基础知识。如果真的要背后的数学感兴趣，我们推荐这本优秀的书：Bruce Schneier 和 Niels Ferguson 所著的《应用密码学》（Practical Cryptography, John Wiley & Sons, 2003）。

使用 TLS 时，数字证书是服务器的公钥，它的作用就像一个电子驾驶执照。它用来识别正在连接的服务器或网站。在连接到一个 HTTPS 网站时，浏览器要做的事就是接受该网站的数字证书作为该网站身份的证据。与驾驶执照一样，数字证书也有使用期限，只在固定的期限内有效，这个期限通常为 12 个月。

每个数字证书还可以包含一个认证中心（Certificate Authority, CA）的证明。CA 是发布证书的机构，它有一个叫根证书的特殊证书用来验证服务器证书的准确性。同样用执照作比喻，根证书就像国家的机动车辆部门。在这里可以验证一个执照是有效的，验证你就是你所谓的你。这些根证书通常与用来连接到服务器的客户端捆绑在一起；例如网络浏览器中就有许多来自著名 CA 的根证书。

那么基于证书的加密（非常简单的用词）基本流程如下所述。

1. 客户端连接到一个服务器，并请求证书。
2. 服务器给出它的证书。
3. 客户端检查根证书的证明。
4. 客户端使用与它捆绑的根证书验证用户的证书是真实的。
5. 如果客户端信任该证书，一条连接就发起了，并在客户端和服务端之间加密。

■提示：在某些情况下，客户端会说它不确定是否信任该证书，并提示你决定是否信任该服务器。

有 4 类的证书需要知道，并且使用每类证书都有正反两方面。

- 由商业 CA 发布的证书。
- 由非商业 CA 发布的证书。
- 由自我管理的 CA 发布的证书。
- 自签发证书。

商业认证中心的证书

来自商业 CA 的证书由 VeriSign、thawte 或者 Go Daddy 之类的提供者发布。这些证书一

般需要定期付费，例如每一年或每半年付一次。价格根据证书的类型和数量不同而不同。大部分商业 CA 的根证书与客户端捆绑在一起，如浏览器、邮件客户端和其他使用 SSL/TLS 连接的工具等。商业 CA 通常定期进行安全检查，因此它们发布的证书一般认为是安全的。

非商业认证中心的证书

除了商业证书提供者之外，还有少数的非商业证书提供者。这些提供者不收取证书费用，但是相应的它们的根证书没有与许多客户端捆绑。这意味着如果在一个网站使用这些证书，或者使用这些证书保护 SMTP 之类的服务，客户端非常可能警告你不能确定证书是否安全、有效。

克服这个问题的唯一方法就是给客户端手动添加一个非商业 CA 的根证书。如果有很多客户端，这可能会增加很多开销和维护工作。在很多情况下，如一个网站，不能接近客户端，那么这些错误就可能导致有人得到客户机不能验证该证书的消息，进而不信任你的网站。举例来说，对于电子商务网站，使用非商业证书是有问题的。

要了解更多信息，请查阅 CAcert 网站和 wiki: <http://www.cacert.org/> 和 <http://wiki.cacert.org/wiki/>。

自我管理的认证中心的证书

你还可以创建和管理自己的证书。这些证书由自己所创建和管理的认证中心发布。结果就是，这些证书不值钱，此外它们确实有另外的问题。首先，既然它是你自己的 CA，就不能期望别人信任你的证书。这导致第二个问题：可用性。与非商业证书一样，你的 CA 的根证书没有和客户端捆绑。

■注：在非商业 CA 的情况下，至少有少数的客户端和根证书捆绑着。而在自我管理的 CA 的情况下，只有你自己给客户端安装根证书，客户端才有根证书。

因此，当客户端试图验证你的 CA 所提供的证书时，就会产生一个错误消息表示客户端不信任这个 CA。为了克服这个错误，你需要在客户端上安装 CA 的根证书。这种事情可以在自己所管理的客户端上做（例如对内部台式机），但是对其他客户端不可行。

■提示：在这种模式下，你必须自己保护和管理 CA。对于少量证书这并不算过分复杂，但它确实会产生一些问题和风险，我们将陆续探讨。

自签发证书

自签发证书不使用 CA。它们由你签署，因此也不值钱。与自我管理的 CA 产生的证书一样，它们一般也不被信任，并会在客户端上产生一个类似的错误信息。与自我管理的 CA 产生的证书不同的是，你不能通过添加一个根证书去掉这个错误，因为没有可添加到客户端的根证书。自签发证书一般只用于测试，很少用在产品平台上。

选择一个证书类型

最好用的证书是由商业 CA 发布的那些证书。关键问题是它需要成本。在某些案例中，

一个商业 CA 的证书一年要\$1500 到\$1800。这对于只保护电子邮件来说是笔可观的费用。那么，如果不希望有购买证书的开销，我们推荐创建自己的 CA 并发布证书。如果客户端数目有限，你可以轻松地在客户端安装 CA 根证书。下一节将说明如何做到这一点。

■注：一般而言，如果在为需要安全性的外部客户运行一个网站（如电子商务网站），你总该从商业 CA 买一个证书。第 11 章会详述这一点。

2. 为 TLS 创建证书

我们已经知道，为了让 TLS 工作，需要两个证书：服务器证书和认证中心（商业 CA 或者自己的 CA）的根证书。我们先生成第一个服务器证书。第一步是生成一个服务器密钥和一个证书签名请求（certificate signing request, CSR）。不管从商业 CA 还是自我管理的 CA 生成证书，我们都会采取这些步骤。

这个过程创建私钥和 CSR。然后这个 CSR 被提交给一个 CA，在本例中就是我们自己的 CA，但也可以是一个商业 CA。正是这个签名过程让客户端确认了一个服务器证书的身份。

列表 10-7 使用 openssl 命令（它是 OpenSSL 程序的一部分）生成了密钥和请求。OpenSSL 程序是一个开源的 SSL 实现，它使 Linux 和其他操作系统能够使用 SSL 加密和保护程序。

列表 10-7 生成一个服务器密钥和请求

```
$ openssl req -new -newkey rsa:4096 -nodes -keyout mail.example.com.key -out mail.example.com.req
Generating a 4096 bit RSA private key
.....++
.....++
writing new private key to 'mail.example.com.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:AU
State or Province Name (full name) [Berkshire]:Victoria
Locality Name (eg, city) [Newbury]:Melbourne
Organization Name (eg, company) [My Company Ltd]:Example Pty Ltd
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:mail.example.com
Email Address []:postmaster@example.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

列表 10-7 使用 openssl 命令生成了一个 RSA 私钥和一个 CSR。为了生成密钥和 CSR，我们使用了 req 和-new 选项。

`-newkey rsa:4096` 选项生成一个 4096 位长的 RSA 私钥。

■注：密钥长度决定加密的安全程度，但是密钥越长，对数据加密所需的性能开销就越大。因此，网站所使用的大部分密钥都是 1024 位长。在我们的例子中，使用密钥是为了保护电子邮件服务器；我们不介意对服务器的性能有多少影响，因此使用了一个更长、更安全的密钥。

传递 `-nodes` 选项告诉 `openssl` 命令不要给新密钥加密，传递 `-keyout` 选项告诉 `openssl` 命令把密钥写在哪里，这里是写到 `mail.example.com.key` 文件。最后一个选项 `-out` 告诉 `openssl` 命令把请求写在哪里。

■提示：输入 `man req`，你可以看到 `openssl req` 选项的更多资料。

然后你就可以看到新密钥和请求产生的过程。在创建请求的过程中，它会提示输入一些有关新证书的信息。如果不想回答某个问题，可以按 `Enter` 键跳过。它会提示输入两字母的国家代码，例如 `US` 代表美国，`GB` 代表大不列颠，`AU` 代表澳大利亚。

■注：在 http://www.iso.org/iso/english_country_names_and_code_elements 上可以找到国家代码的完整列表。

它还会提示输入州、市、组织名和可选的组织单位。这个数据会在客户端查询你的证书时显示。在这里精确和具体是个好主意——特别是在提交 CSR 让一个商业 CA 签名时。

接下来是最重要的，我们需要为证书指定一个通用名。这必须是要使用该证书的服务器的准确主机名。如果指定了一个不正确的主机名，你就会得到一个服务器和证书名不匹配的误差。在本例中指定了 `mail.example.com`，它是我们的电子邮件服务器的完整合格的主机名。

然后我们指定该证书联系人的电子邮件地址，在本例中就是 `postmaster@example.com`。

最后它会提示输入证书的一些附加属性。不必担心这些，你可以按 `Enter` 键跳过。

这个过程会留下两个文件：`mail.example.com.key` 和 `mail.example.com.req`。我们要保管这两个文件，因为在稍后的过程中会用到它们。

下一步是用一个 CA 为 CSR 签名。如果你打算创建自己的 CA，请看下一节“创建自己的认证中心”。

否则你就需要提供 `mail.example.com.req` 文件的内容。我们把该文件交给商业 CA，然后它会给出一个证书。商业 CA 会给出关于如何提供 CSR 文件的说明。我们通常可以剪切、粘贴 CSR 的内容到一个网页并提交。然后 CA 为它签名并在签了名的证书可以下载时通知你。

然后你就可以拿到这个证书并把它用到 Postfix 安装过程中，稍后的“为 Postfix 配置 TLS”一节会讲解这个安装过程。

3. 创建自己的认证中心

创建自己的 CA 是一个容易的任务。首先，我们创建一个目录以保存 CA 和那个目录下的一些子目录。比如说，我们打算把 CA 暂时放在 `/etc/` 目录里。


```
$ sudo mkdir /etc/CA
$ sudo mkdir /etc/CA/{private,newcerts}
$ sudo chown -R root:root /etc/CA
$ sudo chmod 0700 /etc/CA/private
```

private 目录保存 CA 的私钥，**newcerts** 目录包含 CA 将要签名的每个证书的副本。我们还要确保 **root** 用户拥有所有的目录，并保护 **private** 目录，使它只能由 **root** 用户访问。

接下来我们需要创建一个数据库来保存签过名的证书资料。

```
$ echo '01' | sudo tee /etc/CA/serial
$ sudo touch /etc/CA/index.txt
```

■注：第一行把号码 01 echo 给了 `/etc/CA/serial` 文件。为了做到这一点，我们使用了一个叫作 `tee` 的命令，这个命令可以读标准输出然后写到标准输出和文件中。更多的信息可以在 `tee` 命令的 `man` 页找到。

`serial` 文件从号码 01 开始跟踪由该 CA 发布的上个证书的序列号。每个由该 CA 发布的证书都有一个唯一的序列号。

`index.txt` 文件包含当前由该 CA 管理的证书列表。挨着每个证书都有一个字母，表示那个证书的状态。

- R: 撤回 (Revoked)。
- E: 过期 (Expired)。
- V: 有效 (Valid)。

你还需要 OpenSSL 标准配置文件的一个副本。在大多数发行版上都有一个模板文件。这个叫作 `openssl.cnf` 的文件在 Red Hat 上位于 `/etc/pki/tls/` 目录里，在 Ubuntu 上位于 `/etc/ssl/` 目录里。把这个文件复制到新 CA 目录里。

```
$ sudo cp /etc/ssl/openssl.cnf /etc/CA
```

你需要对 `openssl.cnf` 配置文件做一个小改动，以设置新 CA 的默认配置选项。为了做到这一点，你要在配置文件里找到以下面的内容开头的部分。

```
[ CA_default ]
```

在它的下面就是一个叫作 `dir` 的配置选项。把该选项改为以下内容。

```
dir = .
```

它告诉 OpenSSL 在当前目录里寻找配置和签名所需的目录和文件。这意味着在为一个证书签名时，你必须把工作目录改到 CA 目录，在我们的例子中就是 `/etc/CA`。

■提示：看一看 `openssl.cnf` 文件里对 CA 的默认设置。它应该显示已为 CA 创建的默认目录和文件；在我们的例子中，它们是 `private` 目录和 `serial` 文件。

现在你需要为 CA 创建一个自己签名的证书和一个私钥。

```
$ cd /etc/CA
$ sudo openssl req -new -x509 -newkey rsa:4096 -keyout private/cakey.pem ➡
-out cacert.pem -days 3650 -config ./openssl.cnf
Generating a 4096 bit RSA private key
```



```
.....+++++
....+++++
writing new private key to 'private/cakey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:AU
State or Province Name (full name) [Berkshire]:Victoria
Locality Name (eg, city) [Newbury]:Melbourne
Organization Name (eg, company) [My Company Ltd]:Example Pty Ltd
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:ca.example.com
Email Address []:postmaster@example.com
```

首先，正如所提到的那样，我们把工作目录改到/etc/CA。接下来，我们创建一个密钥，这也是 4096 位长的 RSA 密钥。我们把这个密钥存储在/etc/CA/private/cakey.pem 文件。我们还创建了一个自己签名的证书，因为没有其他 CA 为它签名。我们指定该证书的使用期限为 3650 天（或 10 年）。我们还指定一些其他选项：-x509 和 -extensions v3_ca，前者表示证书是自己签的名，后者说明正在创建一个 CA 证书。最后的选项 -config ./openssl.cnf 告诉 OpenSSL 使用已复制到该目录的那个配置文件。

■提示：为了解其他可用选项，你值得通读 openssl.cnf 配置文件。

它会提示输入 CA 私钥的口令或密码。我们选择一个好密码（见第 4 章）并记住它。每次创建新证书时都需要这个密码。它还会提示输入有关你和你的组织的资料，很像前面创建 CSR 时所看到的那样。

现在有了自己的 CA，你可以用它为证书签名。

■提示：还有一个小程序叫作 TinyCA (<http://tinyca.sm-zone.net/>)，它把这个过程中的很多步骤自动化了，并提供一个小型的图形化 CA 平台。你会发现这个工具比手动的过程更易用。

4. 用自己的认证中心为证书签名

既然创建了自己的 CA，你就可以使用它为证书请求签名了。它取出 CSR，用 CA 为它签名，并输出一个签了名的证书。用这个证书可以为 Postfix 配置 TLS。在列表 10-8 中我们为 CSR 签了名。

列表 10-8 为证书请求签名

```
$ cd /etc/CA
$ sudo openssl ca -out /root/mail.example.com.cert -config ./openssl.cnf ➡
-infiles /root/mail.example.com.req
```



```

Using configuration from ./openssl.cnf
Enter pass phrase for ./private/cakey.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
Serial Number: 1 (0x1)
Validity
    Not Before: Jan 4 07:57:56 2009 GMT
    Not After : Jan 4 07:57:56 2010 GMT
Subject:
    countryName           = AU
    stateOrProvinceName   = Victoria
    organizationName      = Example Pty Ltd
    commonName            = mail.example.com
    emailAddress          = postmaster@example.com
X509v3 extensions:
    X509v3 Basic Constraints:
        CA:FALSE
    Netscape Comment:
        OpenSSL Generated Certificate
    X509v3 Subject Key Identifier:
        6F:A6:48:91:D5:3F:70:7B:E3:E2:AB:E5:F5:41:8A:F4:20:DA:31:6E
    X509v3 Authority Key Identifier:
        keyid:F7:0A:17:47:FA:0D:7B:C4:FA:63:0C:C9:FC:5B:49:1C:D5:C3:FF:1D

Certificate is to be certified until Jan 4 07:57:56 2010 GMT (365 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated

```

列表 10-8 使用 `ca` 选项为请求签名。`-out` 选项指定要输出的已签名证书，`-infile` 选项指定要签名的 CSR（假定 CSR 文件在 `/root` 目录里）。最后一个选项 `-config ./openssl.cnf` 指定配置文件，它包含我们的文件和目录的默认位置。

■ **提示：** 输入 `man ca`，你可以看到 `openssl ca` 命令其他的可用选项。

然后它会提示输入为 CA 的私钥所创建的口令，显示证书资料，然后提示为该证书签名并把它资料写到 CA 的数据库里。对两个问题都回答 `y`，即 `yes`。

在这个过程的最后，就有了一个已签名证书、一个私钥和一个证书请求，它们都位于 `/root` 目录。证书有效期为 1 年（可以使用 `-day` 选项指定一个不同的有效期替换它）。

■ **注：** 要小心记录这 3 个文件。下一节会用到它们。

你可以使用 `openssl` 命令检查证书资料。

```
$ openssl x509 -in /root/mail.example.com.cert -noout -text -purpose | more
```

■ **提示：** 保管好 CSR。如果想更新证书（在我们的例子里是 1 年以后），你只需要使用列表 10-8 里的命令重新为这个请求签名。这意味着不必总是重新创建 CSR。

记住，为了避免证书不被信任，必须在相关的客户机里安装根证书，如用户的邮件客户

端！CAcert 网站里有把它的根证书安装到多种客户端的说明：<http://wiki.cacert.org/wiki/BrowserClients>。要安装你的证书，请把说明中它的证书替换为你的证书。

补充知识：管理自己的认证中心

自己管理一个 CA 面临两个关键的问题：安全性和连续性。如果你的 CA 泄了密，任何人都可以使用它创建证书并为证书签名。另一种情况是，如果你的 CA 被意外删除，你需要从头开始创建新的证书以及管理一个新的根证书。

因此，应该总是注意以下两点。

- 备份你的 CA。
- 确保你的 CA 是安全的。

备份 CA 是容易的，第 13 章会介绍备份。保护你的 CA 难一点。我们推荐把 CA 实际存储在一个隔离的主机上。有的人把所用的膝上电脑或者个人电脑放在一个安全或上锁的房间，并上锁保存（再次说明，确保对它进行备份！）。

5. 为 Postfix 配置 TLS

现在有了一个证书和一个密钥，要么是一个商业证书，要么是自我管理的 CA 签发的证书。你还需要来自商业 CA 或者自我管理的 CA 的根证书。如果使用了一个商业证书，CA 一般会提供一个可下载根证书的链接。否则，CA 的证书应该已经在 `/etc/CA/cacert.pem` 里了。

开始先把文件复制到适合配置 Postfix 的某处。我们打算在 `/etc/postfix` 目录下创建一个叫作 `tls` 的目录以保存证书，并把证书复制进来。

```
$ sudo mkdir /etc/postfix/tls
$ sudo cp /root/mail.example.com.cert /etc/postfix/tls
$ sudo cp /root/mail.example.com.key /etc/postfix/tls
$ sudo cp /etc/CA/cacert.pem /etc/postfix/tls
```

现在证书和密钥都在合适的位置，你需要配置 Postfix 的 `main.cf` 配置文件。从列表 10-9 可以看到需往 `main.cf` 配置文件添加的那些选项，我们将介绍这些设置中的每个选项。

列表 10-9 Postfix 的 TLS 配置

```
smtp_tls_security_level = may
smtpd_tls_security_level = may
smtpd_tls_key_file = /etc/postfix/tls/mail.example.com.key
smtpd_tls_cert_file = /etc/postfix/tls/mail.example.com.cert
smtpd_tls_CAfile = /etc/postfix/tls/cacert.pem
smtpd_tls_loglevel = 1
smtpd_tls_received_header = yes
```

列表 10-9 中的头两个选项 `smtp_tls_security_level` 和 `smtpd_tls_security_level` 规定什么时候使用 TLS。你可以看到它们除了开头的前缀之外看起来很相似。一个以 `smtp` 开头，另一个以 `smtpd` 开头。以 `smtp` 开头的配置选项在 Postfix 给另一个电子邮件服务器发送邮件时使用。以 `smtpd` 开头的选项在 Postfix 接收电子邮件（比如来自客户机的邮件）时使用。通过定义 `smtp_tls_security_level` 和 `smtpd_tls_security_level` 这两个选项，告诉 Postfix 要为可能的进站连接和出站连接加密。

这两个选项的 `may` 值打开了一个叫作机会性 TLS（opportunistic TLS）的模式。这基本上意味着如果远程客户端或服务器支持 TLS，就应该使用 TLS。否则，明文连接是可接受

的。这是一个明智的选择，因为不是所有的客户端和服务端都支持 TLS，要求服务器对连接加密意味着有些电子邮件服务器不能给你发邮件。

接下来的 3 个选项 `smtpd_tls_key_file`、`smtpd_tls_cert_file` 和 `smtpd_tls_CAfile` 指定证书、密钥文件和 CA 证书的位置。

再下面的 2 个选项 `smtpd_tls_loglevel` 和 `smtpd_tls_received_header` 控制连接的信息特性。第一个选项控制 Postfix 将产生的 TLS 连接记录量。在这里指定 0 关闭记录，指定 1 提供基本的记录，而 3 和 4 产生最高级别的记录（不推荐，除非是在检修）。我们推荐在日常操作中指定 1，这会产生一些关于连接和所使用证书的简要信息。

`smtpd_tls_received_header` 选项会把报头添加到电子邮件消息里，这些报头提供一些关于 TLS 连接的信息。

做了这个改动之后，你要重启 Postfix 服务。

```
$ sudo service postfix restart
```

重启之后，你可以使用较早前介绍的 `nc` 命令测试，看看 TLS 是否已打开，如列表 10-10 所示。

列表 10-10 测试支持 TLS 的 Postfix

```
$ nc mail.example.com 25
220 mail.example.com ESMTP Postfix
EHLO example.com
250-mail.example.com
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
STARTTLS
220 2.0.0 Ready to start TLS
```

在列表 10-10 中，我们连接到电子邮件服务器并发出了 `EHLO` 命令。邮件服务器用所支持的可用命令回应。你可以看到这里列出了一个新命令 `STARTTLS`。它告诉我们 Postfix 现在为客户端和其他服务器提供 TLS。在列表 10-10 中，接着输入 `STARTTLS` 命令告诉 Postfix 希望发起一个加密的连接。Postfix 回应说它已准备好开启一个加密的连接。这表示 TLS 配置成功，正在等候连接。

如果在配置 Postfix 的 TLS/SSL 加密方面有问题，请看即将介绍的“获得 Postfix 相关的帮助”一节中的建议和链接。

注：为了确保电子邮件内容的机密性，重点要注意使用 Postfix 的 TLS 加密时的几个问题。Postfix 给其他服务器发送电子邮件，但只在另一个服务器支持 TLS 时才为它加密。因为不是所有的服务器都支持加密，所以有些电子邮件不会被加密。要确保所有的电子邮件都加密，唯一的方式是使用像 S/MIME (<http://en.wikipedia.org/wiki/S/MIME>)、PGP (<http://www.pgp.com/>) 和 GnuPG (<http://www.gnupg.org/>) 这样基于内容的加密方案。

10.3.2 身份验证

既然已经配置了 TLS，现在就可以加密客户端与服务器之间以及服务器和其他支持 TLS 的服务器之间的会话。这把我们带到 Postfix 配置的下一个阶段：身份验证。

在默认配置中，Postfix 服务器只接受来自 mynetworks 配置选项所定义的可信任网络的电子邮件，在我们的例子里就是 192.168.0.0/24 和 192.168.1.0/24 网络。这阻止了不适当的用户使用邮件服务器。没有这些限制的电子邮件服务器被称为开放式中继（open relay）。开放式中继允许任何人给它发邮件，服务器会转发它。垃圾邮件发送者广泛利用开放式中继，用不受欢迎的电子邮件污染 Internet。

■注：开放式中继是很有问题的，有些不正确的配置会导致服务器变成一个开放式中继，这对于你的组织是很大的麻烦。当检测到开放式中继时，通常都把它们列入黑名单。这种黑名单的方式意思就是这个开放式中继的 IP 地址被添加到一个服务器列表，而来自这些服务器的电子邮件都不被接受。如果你的服务器是一个开放式中继，甚至在关闭中继并修好服务器之后，也很难从黑名单里移除以允许你的用户发送电子邮件。应该使用像 <http://www.abuse.net/relay.html> 这样的服务定期测试服务器有没有开放式中继的行为。

默认情况下，Postfix 的配置阻止服务器的开放式中继行为。但是尽管该配置阻止了开放式中继，它还是留下一个安全漏洞，产生了一个功能性缺口。

由于这个漏洞，任何得到你网络上某个 IP 地址的人都可以给你的邮件服务器发送电子邮件。比如说，如果攻击者损害了你的无线网络，他就可能利用你的邮件服务器发送垃圾邮件。

该配置也会给移动用户带来功能性缺口。你几乎一定有移动的、在家工作的或者有手机或 PDA 这类移动设备的用户。这些用户不能使用你的邮件服务器，因为是移动的，它们没有你网络上的 IP 地址。它们必须依赖一个允许中继的服务提供者的电子邮件服务器、开放式中继或者 VPN 进入你的网络。通常这不是理想状态。

有了身份验证，用户就可以从任何地方发送电子邮件，只要它们通过了验证。这也意味着所信任的网络上的内部用户在被允许发送电子邮件之前需要提供身份验证信息。这样就降低了有人只是跳到你的网络就能使用邮件服务器的风险。

身份验证反过来受到刚配置的 TLS 加密的保护，从而允许用户通过验证而他们的身份验证信息没有通过网络暴露的潜在危险。

■注：这是一件好事，因为使用 tcpdump 或 Wireshark（两者都在第 6 章提到过）之类的工具探听网络的事情非常普遍。这些人探听加密或暴露的密码之类内容以盗取信息。对于无线网络，这特别容易，因为攻击者甚至不需要物理上插入你的网络。

1. SMTP AUTH 与 SASL

电子邮件服务器的身份验证由一个叫作 SMTP AUTH 的机制提供。这是另一个 SMTP 命令，它在允许用户发送电子邮件之前提示他输入用户名和密码。这个用户名和密码可以通过

多种机制提供，包括明文的和加密的形式。你也可以利用支持一次性密码的机制，如智能卡或令牌。使用什么样的机制取决于你的客户端支持什么机制。比如，Microsoft Outlook 客户端只支持很少的机制，而 Mozilla Thunderbird 支持更广泛的机制。

■注：在 http://en.wikipedia.org/wiki/One-time_password 可以学习一次性密码的知识。

为了确认用户的凭证有效，AUTH 命令使用一个叫作简单鉴定与安全层（Simple Authentication and Security Layer, SASL）的鉴定框架。SASL 很像 PAM（第 4 章讲过），它为身份验证做摘要。它允许多种类型的身份验证隐藏在 SASL 协议后面。这意味着，邮件服务器可以检查多种后端服务以验证用户被允许发送邮件，而无需理解对这些服务是如何做身份验证的。

这些后端服务可以包括 PAM（它可以用来允许用户用他们的 Unix 登录和密码做身份验证）、用户和密码数据库，甚至包括像 LDAP 或 Active Directory 这样的用户库。Postfix 没有内置 SASL。它靠集成其他程序来提供 SASL 功能。我们将使用 Dovecot 服务器为 Postfix 提供这些 SASL 功能。

如果到现在为止你一直照着例子做，那么应该在本章前面说明如何安装 Dovecot 时就已经安装了 Dovecot。如果没有安装但是想完成下面的指令，那么你现在就使用那些指令安装它。然后你要确保 Postfix 支持用 Dovecot 做 SASL 身份验证。使用 `postconf` 命令和 `-a` 选项可以完成此事。

```
$ sudo postconf -a
cyrus
dovecot
```

在这里 `postconf` 命令返回了 Postfix 支持的所有 SASL 身份验证插件。我们在寻找 `dovecot` 项。Red Hat 和 Ubuntu 近来所有的版本都支持 `dovecot`。

2. 为 Dovecot 配置 SASL

现在需要配置 Dovecot 的 SASL 功能并开启 Dovecot 后台程序。Dovecot 配置文件在 Red Hat 上位于 `/etc/dovecot.conf`，在 Ubuntu 上位于 `/etc/dovecot/dovecot.conf`。这两个文件都是在安装 Dovecot 软件包时创建的，并且包含配置样例，其中包括了很多种归档选项。

我们将编辑该文件并设置 Dovecot 的 SASL 身份验证服务。打开这个配置文件，找到要编辑的第一个选项：`protocols`。首先我们希望关闭所有非身份验证的服务。为此，我们把它设置为 `none`。

```
protocols = none
```

■注：在本章稍后“配置 IMAP 和 POP3”一节讲述 IMAP 和 POP3 以及如何启动它们时会回到这个选项。

接下来我们需要配置身份验证（或 `auth`）服务。为做此事，我们在 `dovecot.conf` 配置文件里找到 `auth default` 配置选项。

从列表 10-11 可以看到要设置的 auth 服务配置选项。

列表 10-11 配置 Dovecot 的 auth 服务

```
auth default {
    mechanisms = plain login
    passdb pam {
    }
    userdb passwd {
    }
    socket listen {
        client {
            path = /var/spool/postfix/private/auth
            mode = 0660
            user = postfix
            group = postfix
        }
    }
}
```

auth 服务由包含在大括号（{ }）内的一连串指令组成。该服务可能在已有的 dovecot.conf 配置文件里已经配置一部分了。你要确保已存在的配置文件里的配置与列表 10-11 相符。

让我们看看每一个选项。auth 服务的第一个指令是 mechanisms，它规定这个 Dovecot 例程支持哪些身份验证机制。默认情况下，接受明文形式的用户和密码的 PLAIN 机制通常总是打开。

■注意：如果打开了 TLS，你应该只使用 PLAIN 机制；否则攻击者会从网络窃取用户的凭证。

从表 10-1 可以看到 Dovecot 的其他可用类型。

表 10-1 Dovecot 的身份验证机制	
机制	说明
PLAIN	明文身份验证
LOGIN	一种在 Microsoft Outlook 客户端使用的 Microsoft 身份验证机制
CRAM-MD5	加密的密码机制。在邮件客户端有一些支持
DIGEST-MD5	与 CRAM-MD5 类似，但密码更强。它在客户端只有有限的支持
NTLM	基于 Microsoft Windows 的身份验证，一般只有 Microsoft 客户端支持
GSSAPI	Kerberos v5 的配套机制。在客户端的支持有限
ANONYMOUS	支持匿名登录。不推荐而且不安全
OTP	一次性密码机制
SKEY	一次性密码机制

■注：你不能指定一个 Dovecot 的配置所不支持的身份验证机制。例如，在没有恰当的配套配置的情况下，你不能指定 NTML 机制。

在列表 10-11 中还打开了 LOGIN 身份验证机制，以防用户中有 Microsoft 客户端，但我

们不打算打开其他任何类型。绝大多数客户端都支持 PLAIN，其他一些客户端也支持 LOGIN 身份验证类型。在 http://en.wikipedia.org/wiki/Comparison_of_e-mail_client#Authentication_support 上可以找到邮件客户端和它们的身份验证机制的完整列表。

列表 10-11 中接下来的指令控制 Dovecot 为执行身份验证而用来查验的仓库。我们定义了一个身份验证仓库：passdb pam。这个 passdb pam 仓库是一个密码数据库，它利用 PAM 程序为不同于本地主机用户的用户做身份验证。因此对于为发送电子邮件而做身份验证的用户，需要在这个主机上创建一个带有效密码的用户。

在验证用户时，Dovecot 从/etc/pam.d 目录寻找一个叫作 dovecot 的 PAM 服务定义。这个文件是在安装 Dovecot 软件包时安装的。列表 10-12 显示了 Ubuntu 里的/etc/pam.d/dovecot 文件。

列表 10-12 Dovecot 的 PAM 服务

```
#%PAM-1.0
@include common-auth
@include common-account
@include common-session
```

如列表 10-12 所示，对 Dovecot 的身份验证查询所使用的 PAM 认证校验与用户登录该主机所经历的校验（第 4 章介绍过）是一样的。

■注：Dovecot 密码数据库的更多信息见 <http://wiki.dovecot.org/PasswordDatabase>，Dovecot PAM 身份验证的更多信息见 <http://wiki.dovecot.org/PasswordDatabase/PAM>。

列表 10-11 中接下来的指令 userdb passwd 执行一个用户查找以返回一些与该用户相关的信息。它返回该用户的 UID、GID 和主目录。这个信息是从/etc/passwd 文件检索到的。

■注：你可以到 <http://wiki.dovecot.org/UserDatabase> 阅读更多有关用户数据库查找的内容。

列表 10-11 中最后一个指令是 socket 指令，它提供 Postfix 和 Dovecot 之间的连接。这个套接字监听来自 Postfix 的身份验证请求，然后返回结果。它使用第 3 章简要介绍过的被称作套接字的特殊文件类型来做此事，它允许程序之间进行交互。这个套接字有一个文件名 auth，并且位于一个 Postfix 目录/var/spool/postfix/private/，此目录存储这个后台程序所使用的文件与套接字。mode、user 和 group 选项控制这个套接字的权限和所有权。

在 Dovecot 后台程序运行时，你可以在这个目录里看到这个文件。

```
$ ls -l /var/spool/postfix/private/auth
srw-rw---- 1 postfix postfix 0 Jan 7 18:37 auth
```

配置 Dovecot 之后，你需要启动（或重启）它。对于 Red Hat，我们使用 service 命令打开 Dovecot。

```
$ sudo service dovecot start
```

对于 Ubuntu，我们使用 invoke-rc.d 命令。

```
$ sudo invoke-rc.d dovecot start
```


你可以通过系统日志输出确认 Dovecot 正在运行，系统日志输出通常在 Red Hat 上的 `/var/log/maillog` 文件里或者 Ubuntu 上的 `/var/log/mail.log` 文件里。你应该像下面这样查看输入项。

```
dovecot: Jan 19 12:29:46 Info: Dovecot v1.1.7 starting up
```

或者，你可以验证 Dovecot 进程正在运行。

```
$ ps -A | grep 'dovecot'
7331 ?          00:00:00 dovecot
7333 ?          00:00:00 dovecot-auth
```

在这里你可以看到 `dovecot` 和 `dovecot-auth` 进程正在运行，表示 Dovecot 已经成功开启。

在 Red Hat 上，Dovecot 的信息记录到 `/var/log/maillog` 文件，而在 Ubuntu 上则记录到 `/var/log/mail.log` 文件，如果看到一个如下这样的日志输入项，就可以确认它在运行。

```
Jan 7 18:37:03 au-mel-rhel-1 dovecot: Dovecot v1.0.7 starting up
```

3. 为 Postfix 配置 SASL

接下来我们需要配置 Postfix，使它可以使用刚配置过的 Dovecot 的 SASL 服务。把列表 10-13 中的输入项添加到 `main.cf` 配置文件。

列表 10-13 为 Postfix 配置 Dovecot 的 SASL 服务

```
smtpd_sasl_type = dovecot
smtpd_sasl_path = private/auth
smtpd_sasl_auth_enable = yes
smtpd_tls_auth_only = yes
smtpd_recipient_restrictions = permit_mynetworks, ➡
permit_sasl_authenticated, reject_unauth_destination
```

在列表 10-13 中，我们用 `smtpd_sasl_type` 选项确定要使用 Dovecot 执行 SASL 身份验证功能。`smtpd_sasl_path` 选项指定身份验证套接字在 Postfix 的 `spool` 目录（通常为 `/var/spool/postfix` 目录）中的位置。这和前面在列表 10-11 中定义的 Dovecot 客户端套接字相符。`smtpd_sasl_auth_enable` 选项告诉 Postfix 打开 SASL 身份验证功能。

下一个选项 `smtpd_tls_auth_only` 告诉 Postfix 如果 TLS 已经打开并正在运行的话，就只使用身份验证。这意味着 `STARTTLS` 命令必须已经发出，而且已经在客户端和服务器之间创建一条加密连接。

最后一个选项 `smtpd_recipient_restrictions` 是 Postfix 约束列表之一。它告诉 Postfix 当 `RCPT TO` 命令发出时（比如在从客户端接收到一封电子邮件时）要允许或拒绝的内容。如前所述，在默认情况下 Postfix 会接受下面这些位置的电子邮件。

- 发自 IP 地址与 `mynetworks` 选项里的值相匹配的客户端。
- 发往与 `relay_domains` 选项值相匹配的远程主机，这个选项的默认值与 `mydestination` 选项的值相同。
- 发往或者发自本地主机。

我们将调整这个默认行为，告诉 Postfix 也接受来自 SASL 验证用户的电子邮件。

那么我们首先使用 `permit_mynetworks` 选项，该选项保留了 `mynetworks` 选项里那些网络

的访问。然后我们添加 `permit_sasl_authenticated` 选项，告诉 Postfix 接受 SASL 验证过的用户的电子邮件。最后，为了让 Postfix 接收电子邮件并有一个有效的配置，我们必须用一个拒绝约束结束，在本例中就是 `reject_unauth_destination` 选项。这个选项拒绝任何与刚确定的最后两个条件（发往指定的远程目的地或者发自/发往本地主机）不相符的电子邮件。

■注：在 http://www.postfix.org/postconf.5.html#smtpd_recipient_restrictions 上可以看到更多关于收件人约束的信息。

4. 测试 Postfix 的身份验证功能

现在如果重启 Postfix，SASL 配置应该可以起作用，这一点可以测试。测试方式有很多。第一种方式是配置一个客户端让它向服务器发送经过验证的电子邮件。但是因为还没有给客户端设置浏览邮箱的方式（如 IMAP 或 POP3 协议），所以这种方式不太有效。我们打算改用另一种方法，利用一个叫作 `swaks` 的新命令来测试身份验证功能。

`swaks` 命令是比测试 SMTP 服务器的 `nc` 命令更先进的版本。`swaks` 工具可以测试许多 SMTP 选项，包括加密和身份验证，而这对 `nc` 来说很难测试。你可以从 <http://jetmore.org/john/code/#swaks> 下载它，或者利用 Ubuntu 上的可用软件包。

我们在 Ubuntu 上安装 `swaks` 软件包。

```
$ sudo aptitude install swaks
```

Red Hat 没有现成的软件包，但你可以从它的网站下载这个工具。

```
$ cd /usr/local/bin
$ sudo wget http://jetmore.org/john/code/swaks
$ sudo chmod 0755 /usr/local/bin/swaks
```

`swaks` 命令非常易用，你应该阅读它的 `man` 页，寻找有关如何充分利用它的资料。

为了测试身份验证功能，我们需要知道主机上有效用户的用户名和密码，而且要安装 `swaks` 工具所需的一些配套模块以进行身份验证和加密测试。在 Red Hat 上，我们需要下面的软件包。

```
$ sudo yum install perl-Digest-SHA1 perl-Digest-HMAC perl-Net-IP perl-Net-DNS ➡
perl-Net-SSLeay
```

在 Ubuntu 里，我们需要下面这些软件包。

```
$ sudo aptitude install libnet-dns-perl libnet-ssleay-perl libdigest-sha1-perl ➡
libdigest-hmac-perl libnet-ip-perl
```

这些软件包为 `swaks` 提供 SSL/TLS 和 SMTP AUTH 支持，利用它们就可以测试 Postfix 的身份验证功能。

从列表 10-14 可以看到一个测试 SASL 身份验证功能的 `swaks` 会话。

列表 10-14 使用 `swaks` 测试 SASL

```
$ swaks -tls -a -au ataylor -ap password -t jsmith@example.com -f ➡
ataylor@example.com
```


第 10 章 邮件服务

```
=== Trying mail.example.com:25...
=== Connected to mail.example.com.
<- 220 mail.example.com ESMTP Postfix
-> EHLO localhost
<- 250-mail.example.com
<- 250-PIPELINING
<- 250-SIZE 10240000
<- 250-VRIFY
<- 250-ETRN
<- 250-STARTTLS
<- 250-ENHANCEDSTATUSCODES
<- 250-8BITMIME
<- 250 DSN
-> STARTTLS
<- 220 2.0.0 Ready to start TLS
=== TLS started w/ cipher DHE-RSA-AES256-SHA
~> EHLO localhost
<~ 250-mail.example.com
<~ 250-PIPELINING
<~ 250-SIZE 10240000
<~ 250-VRIFY
<~ 250-ETRN
<~ 250-AUTH PLAIN LOGIN
<~ 250-ENHANCEDSTATUSCODES
<~ 250-8BITMIME
<~ 250 DSN
~> AUTH LOGIN
<~ 334 VXNlcm5hbWU6
~> YXRheWxvcg==
<~ 334 UGFzc3dvcmQ6
~> cGFzc3dvcmQ=
<~ 235 2.0.0 Authentication successful
~> MAIL FROM:<ataylor@example.com>
<~ 250 2.1.0 Ok
~> RCPT TO:<jsmith@example.com>
<~ 250 2.1.5 Ok
~> DATA
<~ 354 End data with <CR><LF>.<CR><LF>
~> Date: Wed, 07 Jan 2009 21:11:10 +1100
~> To: jsmith@example.com
~> From: ataylor@example.com
~> Subject: test Wed, 07 Jan 2009 21:11:10 +1100
~> X-Mailer: swaks v20061116.0 jetmore.org/john/code/#swaks
~>
~> This is a test mailing
~>
~> .
<~ 250 2.0.0 Ok: queued as 67972A0B92
~> QUIT
<~ 221 2.0.0 Bye
=== Connection closed with remote host.
```

在列表 10-14 中，我们启动了带-tls 选项的 swaks 命令，这个选项告诉 swaks 使用 TLS 加密。然后我们指定打开身份验证功能的-a 选项和分别指定用户名和密码的-au、-ap 选项。

接下来我们用-t和-f选项分别指定这封电子邮件的来源和目的地。

当该命令被执行时，你可以看到发起了一个连接，启动了 STARTTLS 命令，还有 TLS 已启动的确认信息。在这个 TLS 连接之后所产生的命令列表中，你可以看到可用的 AUTH 命令提供了两种身份验证机制：PLAIN 和 LOGIN。

然后 swaks 命令启动了 AUTH 命令，并提交了用户名和密码。LOGIN 身份验证机制被选定。身份验证成功，测试邮件被提交。如果这个身份验证失败了，Postfix 会一直尝试这些机制直到有一个机制成功或者全都失败，在本例中它先尝试 LOGIN 然后是 PLAIN。如果全失败了，客户端会接收到一个错误消息。

你还可以通过检查 Postfix 的系统日志文件确认会话已成功。

```
$ sudo less /var/log/maillog
Jan 7 21:11:11 au-mel-rhel-1 postfix/smtpd[12638]: ➡
connect from localhost.localdomain[127.0.0.1]
Jan 7 21:11:11 au-mel-rhel-1 postfix/smtpd[12638]: setting up TLS connection from ➡
localhost.localdomain[127.0.0.1]
Jan 7 21:11:11 au-mel-rhel-1 postfix/smtpd[12638]: TLS connection established ➡
from localhost.localdomain[127.0.0.1]: TLSv1 with cipher DHE-RSA-AES256-SHA ➡
(256/256 bits)
Jan 7 21:11:11 au-mel-rhel-1 postfix/smtpd[12638]: 67972A0B92: ➡
client=localhost.localdomain[127.0.0.1], sasl_method=LOGIN, sasl_username=ataylor
Jan 7 21:11:11 au-mel-rhel-1 postfix/cleanup[12643]: 67972A0B92: ➡
message-id=<20090107101111.67972A0B92@au-mel-rhel-1.localdomain>
Jan 7 21:11:11 au-mel-rhel-1 postfix/smtpd[12638]: disconnect from ➡
localhost.localdomain[127.0.0.1]
Jan 7 21:11:11 au-mel-rhel-1 postfix/qmgr[12292]: 67972A0B92:
from=<ataylor@example.com>, size=562, nrcpt=1 (queue active)
```

注意，TLS 连接建立起来了，身份验证功能启动了，电子邮件也发送出去了。

如果身份验证成功，那么现在 Postfix 身份验证功能就在运行，服务器就可以被远程用户使用，内部用户也可以安全地提交电子邮件。

如果在配置 Postfix 身份验证功能方面有问题，请看即将介绍的“获得 Postfix 相关的帮助”一节里的建议和链接。

补充知识：SMARTHOSTING

本章前面提到过，一些 ISP 阻止往外去的 SMTP 流量。这样设计是为了帮助减少垃圾邮件。另外，一些 SMTP 服务器被配置为不接收来自某些类型的网络的电子邮件，如 ADSL 和使用动态 IP 地址的网络。如果不能往外发送电子邮件或者某些服务器拒绝来自服务器的连接，这可能就是问题。检验这一点的一个简单方法是使用 swaks 这样的工具，看看能否连接到一个外部 MTA，或者可以联系 ISP 查明它是否限制往外去的 SMTP 流量。

如果被阻止了，你可以用 smarthosting 克服这个问题。smarthost 是接收并转发电子邮件的中继服务器。许多 ISP 都为它们的用户提供 smarthost。

有两类 smarthost——不需身份验证的和需要身份验证的。两种类型都是用 main.cf 配置文件里的 relayhost 配置选项指定（见 <http://www.postfix.org/postconf.5.html#relayhost>）。如下所示（其中 mail.isp.net 是 smarthost 的主机名）。

```
relayhost = mail.isp.net
```

它告诉 Postfix 邮件服务器把所有往外发的电子邮件发送给 mail.isp.net。它假定 smarthost

会接受来自你的电子邮件；例如，一些 ISP 的 smarthost 乐意接受来自本 ISP 所管辖 IP 地址的电子邮件。

然而，有时在没有 SASL 身份验证机制的情况下，smarthost 不接受来自你的电子邮件。如果是这样的话，你就需要配置 smarthost 的身份验证功能。为此，你要配置 SASL 身份验证功能，你在这里 Postfix 是客户端而不是服务器。记住，当把它当作服务器配置时，所有的配置选项以 smtpd_ 开头，表示它是为进入该服务器的连接而配置的。现在你需要把 Postfix 配置为一个客户端，因此要使用以 smtp 开头的配置选项。为此，我们在 main.cf 文件里设置下面的选项。

```
smtp_sasl_password_maps = hash:/etc/postfix/smtp_sasl_passwd
smtp_sasl_auth_enable = yes
smtp_sasl_mechanism_filter = plain, login
smtp_sasl_security_options = noanonymous
```

这就为作为客户端的服务器配置了 SASL 身份验证功能。smtp_sasl_password_maps 选项（见 http://www.postfix.org/postcnf.5.html#smtp_sasl_password_maps）指定一个包含 smarthost 和它们所需凭证列表的数据库文件。这是通过编辑文件并使用 postmap 命令从那个文件创建一个数据库而创建的。/etc/postfix/smtp_sasl_passwd 文件需包含 smarthost 的名字和身份验证所需的用户名和密码（用冒号隔开），如下所示。

```
mail.isp.net username:password
```

你要确保这些文件只对 root 用户可读，并用适当的权限阻止任何人查看密码。

```
$ sudo chown root:root /etc/postfix/smtp_sasl_passwd
$ sudo chmod 0600 /etc/postfix/smtp_sasl_passwd
```

我们接着使用 postmap 命令创建一个数据库。

```
$ sudo postmap hash:/etc/postfix/smtp_sasl_passwd
```

然后一个叫作 smtp_sasl_passwd.db 的文件就被创建了。

剩下的选项都相当简单。smtp_sasl_auth_enable 选项为作为客户端的 Postfix 打开 SASL 身份验证功能。smtp_sasl_mechanism_filter 选项指定该 smarthost 所支持的身份验证机制类型，smtp_sasl_security_options 选项关掉 anonymous 机制。

配置更新之后，你要重新载入或重启 Postfix，然后可以测试服务器能否给 smarthost 发送 SASL 验证过的电子邮件。你还可以用传送选项指定更多粒度的 smarthosting（见 <http://www.postfix.org/transport.5.html>）。

10.4 获得 Postfix 相关的帮助

Postfix 不仅是一个有简单配置模式和许多文档、配置最容易的程序之一，还是提供很多有用的描述性错误消息、检修最友好的程序之一。

■注：不是所有的 MTA 都这么容易配置，其他 MTA 有复杂的语法和令人困惑的错误消息。你会发现 Postfix 比某些可选 MTA 更容易理解。

你可以在 Postfix 主页（<http://www.postfix.org/>）上找到有用的文档。它包括文档（<http://www.postfix.org/docs.html>）。

postfix.org/documentation.html)、操作指导和 FAQs (<http://www.postfix.org/docs.html>)。

特别是在配置和检修 SSL/TLS 加密和 SASL 功能方面,有一些有用的资源。对于加密,你可以在 http://postfix.org/TLS_README.html 上找到资源。对于 SASL,你可以在 http://www.postfix.org/SASL_README.html 上找到有用的操作指导,另一个针对 Ubuntu 的资源在 <http://adomas.org/2006/08/postfix-dovecot/>, 并且更高级的版本在 http://www.lxtreme.nl/index.pl/docs/linux/dovecot_postfix_pam。

你还可以在 http://www.postfix.org/SOHO_README.html 上找到有关在小型办公室/家庭办公环境里运行 Postfix 的一些提示,另外,你也许要考虑加入 <http://www.postfix.org/lists.html> 上的 Postfix 邮寄列表。

记住,如果要提交问题或程序错误,应该包含下面的信息。

- Postfix 配置 (运行 `postconf -n`)。
- 平台 (运行 `uname -a`)。
- 所产生的全部日志消息 (在 Ubuntu 上的 `/var/log/mail.log` 文件里,或者在 Red Hat 上的 `/var/log/maillog` 文件里)。

另外一个可以利用的方式是 Freenode IRC 服务器 (<http://freenode.net/>) 上叫作 #postfix 的 IRC 通道,你可以在那里寻求帮助。

10.5 防止病毒和垃圾邮件

既然 Postfix 已经可以运行加密和身份验证功能,那么下面就介绍如何保护用户和组织免受垃圾邮件和病毒的危害。我们将研究下面两个工具。

- SpamAssassin: 一个开源的反垃圾邮件工具。
- ClamAV: 一个开源的反病毒扫描器和引擎。

我们将把这两个工具与 Postfix 集成在一起并讲解它们的使用方法。

10.5.1 与垃圾邮件作战

垃圾邮件是主动提供的电子邮件,范围从帮助无数非法财务往来的请求到扩大人体解剖组织的医学捐献。垃圾邮件是用户幸福的最大威胁之一。你要工作时却发现大量垃圾邮件而真正邮件被埋没,没有什么比这更气人的了。对组织和用户来说,它还是一个威胁,因为垃圾邮件往往隐藏着网络钓鱼攻击 (<http://en.wikipedia.org/wiki/Phishing>)、病毒散布和其他类型的恶意攻击。

我们将配置 Postfix 服务器主动拒绝一些垃圾邮件,然后引入一个叫作 SpamAssassin 的流行反垃圾邮件工具。SpamAssassin 是一个 Bayesian 垃圾邮件过滤器 (http://en.wikipedia.org/wiki/Bayesian_spam_filtering)。用最简单的话说,邮件中出现某个词、词组或其他特征可能意味着该邮件为垃圾邮件,而 Bayesian 垃圾邮件过滤 (Bayesian spam filtering) 就是预测这种可能性的一种方法。

每封电子邮件根据一连串的可定制测试或规则计算出一个量化分值，并按照这个分值被标记为垃圾邮件或非垃圾邮件；默认情况下，超过 5.0 分就标记为垃圾邮件。每个规则根据它的权重加分或减分。例如，一个规则可能检查电子邮件的某个具体特征，如果它与这个特征相符，SpamAssassin 可能就把 0.5 分加到这封电子邮件的分数上。

Bayesian 垃圾邮件过滤器还能从用户所输入的邮件学习一些模式，可以通过被告知哪个邮件是垃圾邮件（spam）、哪个不是（被 SpamAssassin 称作 ham）而得到训练（[http://en.wikipedia.org/wiki/Spam_\(food\)](http://en.wikipedia.org/wiki/Spam_(food))）。它学到的数据被添加到一个数据库，用于将来对输入的邮件做更准确的分析。

SpamAssassin 以主机后台程序的形式运行。电子邮件被提交给这个后台程序、被分析，然后通过添加一个新报头返回标记为是或不是垃圾邮件的结果。

1. 为 Postfix 配置反垃圾邮件功能

在配置 SpamAssassin 之前，要收紧 Postfix 配置。为此，我们往 main.cf 配置文件添加一些配置选项。其中的大部分选项都拒绝与 SMTP RFC 不兼容的电子邮件，比如拒绝源地址非有效邮件的电子邮件。

我们主要更新在研究 Postfix 身份验证时所介绍的约束列表。表 10-2 给出了这些约束列表。

表 10-2 Postfix 的约束列表	
约束列表	说明
smtpd_client_restrictions	客户端连接时的约束
smtpd_helo_restrictions	HELO/EHLO 命令发出时的约束
smtpd_sender_restrictions	MAIL FROM 命令发出时的约束
smtpd_recipient_restrictions	RCPT TO 命令发出时的约束
smtpd_data_restrictions	DATA 命令发出时的约束

约束列表在特定事件发生时被触发；例如，当客户端连接时检查 smtpd_client_restrictions，当 EHLO 命令由客户端发出时检查 smtpd_helo_restrictions。

在本例中，我们要给发送人、收件人和数据约束列表添加一些选项。首先，我们往 main.cf 文件添加 smtpd_sender_restrictions 选项，如下所示。

```
smtpd_sender_restrictions = reject_non_fqdn_sender, reject_unknown_sender_domain
```

reject_non_fqdn_sender 选项拒绝那些发送人邮寄地址格式不正确的电子邮件。这意味着 MAIL FROM 命令传递的值必须是有效电子邮件的地址格式，如 ataylor@example.com。像 Anne Taylor、ataylor 或 Anne 以及类似的不能回应的地址不会被接受，因此该邮件会被丢弃。

reject_unknown_sender_domain 选项拒绝那些发件人域里没有 DNS 的 A 记录或 MX 记录的电子邮件。这通常都是通过伪造的域发送的电子邮件，垃圾邮件往往是这种情况。

我们接下来要给 smtpd_recipient_restrictions 选项添加另一个拒绝标准。

```
smtpd_recipient_restrictions = permit_mynetworks, permit_sasl_authenticated, ➡  
reject_unauth_destination, reject_unauth_pipelining
```


这里更新了已有的选项（本章早前给 Postfix 添加身份验证功能时所配置的选项），从而把 `reject_unauth_pipelining` 约束包含进来，此约束拒绝那些使用一个叫作 `pipelining` 的技术而没有检查是否支持 `pipelining` 就提交的电子邮件。这是垃圾邮件邮寄者用来提交电子邮件的常见技术。

同样的选项还要添加到 `smtpd_data_restrictions`，以便抓住那些在 `DATA` 命令发出时使用同样技术的垃圾邮件邮寄者。

```
smtpd_data_restrictions = reject_unauth_pipelining
```

最后，我们要配置两个与约束列表无关但阻止某些垃圾邮件的选项。

```
smtpd_helo_required = yes
disable_vrfy_command = yes
```

第一个选项 `smtpd_helo_required` 告诉 Postfix 拒绝来自没有发送正确 EHLO 以及没有通告名字的那些客户端的连接。`disable_vrfy_command` 选项关闭 SMTP 的 VRFY 命令。VRFY 命令允许发送人查询 Postfix 服务器并确认某个地址是否存在。这被垃圾邮件邮寄者用来验证地址，偶尔还会被黑客用来在攻击前收集主机上的用户名。

■注：还有其他的约束可以打开，但这些是最简单、最容易和最不可能限制到 Postfix 服务器合法邮件的约束。更多相关信息见 <http://www.postfix.org/uce.html>。

做完改动后，我们重启 Postfix 后台程序。

■注：做改动后再确认一下收发电子邮件的功能是个好主意。

2. 安装与配置 SpamAssassin

为了补充对 Postfix 的改动，将安装和配置 SpamAssassin。从安装必需的软件包开始。在 Red Hat 上，我们需要安装 `spamassassin` 软件包并告诉它自动开启 SpamAssassin 后台程序。

```
$ sudo yum install spamassassin
$ sudo chkconfig spamassassin on
```

在 Ubuntu 上，我们需要安装 `spamassassin` 和 `spamc` 软件包。

```
$ sudo aptitude install spamassassin spamc
```

然后我们开启 SpamAssassin 后台程序，方法是编辑 `/etc/default/spamassassin` 文件并把 `ENABLED=0` 选项设置为下面的内容。

```
ENABLED=1
```

■注：在 Red Hat 和 Ubuntu 上安装 SpamAssassin 软件包时，有一些作为先决条件的附加软件包也要安装。

然后我们启动 SpamAssassin 的 `spamd` 后台程序。在 Red Hat 上，我们使用 `service` 命令。

```
$ sudo service spamassassin start
```


对于 Ubuntu，我们使用 `invoke-rc.d` 命令。

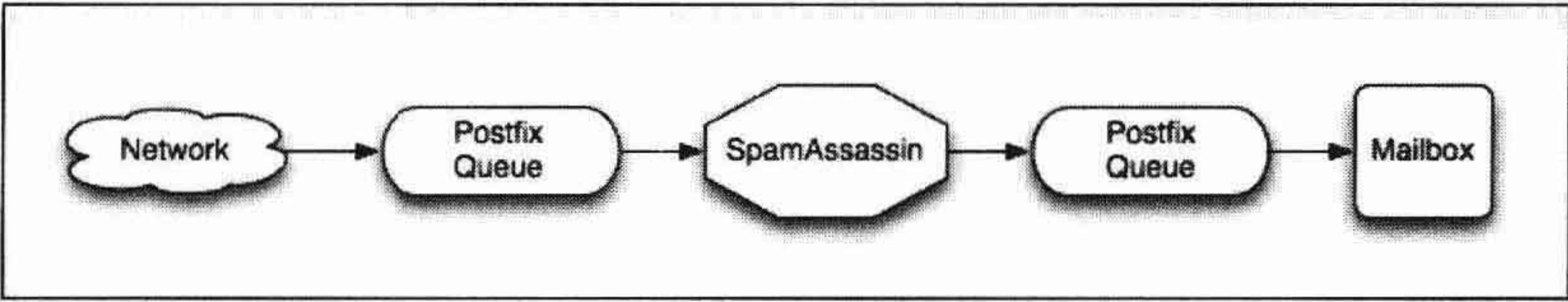
```
$ sudo invoke-rc.d spamassassin start
```

■提示：与 Postfix 一样，对配置做完改动之后重启 SpamAssassin 是一个好主意。

3. 为 Postfix 配置 SpamAssassin

在邮件平台上集成像 SpamAssassin 这样的垃圾邮件过滤器有许多不同的方式。一种方式是在把邮件投递给用户之前使用一个 MDA 扫描用户的来信。然后垃圾邮件过滤器通常往邮件里添加一个报头，MDA 检测到这个报头并把垃圾邮件导向一个特别的文件夹。

另一个方法是当邮件还在 MTA 里时就对它进行分析。这样可以把邮件从 Postfix 的邮件队列传递给 SpamAssassin。然后邮件被扫描，再发回给 Postfix 投递。接下来，客户端或者 MDA 可以利用扫描的结果以及添加到邮件的报头，决定对此邮件做什么处理。从图 10-6 可以看到所提到的电子邮件通过 Postfix 的流程。



Network: 网络; Postfix Queue: Postfix 队列; Mailbox: 邮箱
图 10-6 SpamAssassin 和 Postfix 过滤流程

我们打算选择后一种方法，因为它对大多数平台更高效、更有伸缩性。

首先，我们更新 `master.cf` 文件以便把邮件发送给 SpamAssassin。为此，我们调整 `master.cf` 文件里的 `smtp` 服务，如下所示。

```
smtp inet n - n - - smtpd
-o content_filter=spamassassin
```

这里添加了这一行 `-o content_filter=spamassassin` (`-o` 用空格缩进是为了表示它接续上一行)。`content_filter` 选项告诉 Postfix 要把所有投递给 `smtp` 服务的电子邮件发送给一个叫 `spamassassin` 的过滤器。

现在我们需要定义这个过滤器。为此，我们把该过滤器添加到 `master.cf` 文件的底部。如下这样定义它。

```
spamassassin unix - n n - - pipe
user=mail argv=/usr/bin/spamc -e /usr/sbin/sendmail -oi -f ${sender} ${recipient}
```

这样就在 `master.cf` 文件里创建了一个类型为 `unix` 的新服务。这是一个 Unix 套接字。它调用另外一个叫作 `pipe` 的 Postfix 后台程序，这个后台程序把电子邮件投递给一个外部命令。在接下来的行里（也是缩进的）指定了那个外部命令。

我们指定运行该命令的用户为 `mail`，然后是要传递给该命令的命令和参数。我们在调用 `spamc` 命令。这个二进制代码连接到 `spamassassin` 后台程序，提交电子邮件，然后接收

扫描结果。

我们给 `spamc` 命令传递 `-e` 参数。这个参数（必须在命令行的最后指定）告诉 `spamc` 命令对扫描后的电子邮件做什么处理。本例中把它提交给了 `/usr/sbin/sendmail` 命令，这个 Postfix 命令把邮件投回 Postfix 以待投递给用户。

我们还为 `sendmail` 命令指定了一些选项。`-oi` 选项告诉 `sendmail` 命令当它发现有单个句点的行时不要停止处理邮件（记得前面测试邮件时会发送一个带句点的行作为结尾）。这是因为邮件某行有一个句点可能不表示邮件的结束。`-f ${sender}` 选项确保邮件的发送人被发送给 Postfix。最后，`${recipient}` 选项包含邮件的收件人，以使 Postfix 知道把该邮件发送给谁。

4. 测试 SpamAssassin

为了测试 SpamAssassin，让我们使用 `swaks` 工具发送一封电子邮件，然后分析它的结果。

```
$ swaks -tls -a -au ataylor -ap password -t jsmith@example.com -f ➡
ataylor@example.com --body /usr/share/doc/spamassassin-version/sample-spam.txt
```

该命令连接到邮件服务器并从 SASL 验证过的用户 `ataylor` 那里给用户 `jsmith@example.com` 发送了一封电子邮件。我们还添加了 `--body` 选项来指定邮件的内容。本例使用 SpamAssassin 软件包自带的一个垃圾邮件实例。这封电子邮件位于 `/usr/share/doc/spamassassin-version` 目录（用 SpamAssassin 版本号替换 `version`，如 3.2.5）。这个垃圾邮件实例保证触发 SpamAssassin 的垃圾邮件过滤器，从而可以测试垃圾邮件检测功能。

如果检查 Postfix 日志，你就可以看到这封电子邮件已被接收和处理，如列表 10-15 所示。

列表 10-15 与 SpamAssassin 相关的 Postfix 日志

```
$ sudo less /var/log/mail.log
Jan 10 10:43:29 au-mel-rhel-1 postfix/smtpd[9599]: connect from ➡
unknown[192.168.0.128]
Jan 10 10:43:29 au-mel-rhel-1 postfix/smtpd[9599]: A759D207EC: ➡
client=unknown[192.168.0.128], sasl_method=LOGIN, sasl_username=ataylor
Jan 10 10:43:29 au-mel-rhel-1 postfix/cleanup[9608]: A759D207EC: ➡
message-id=<20090109234329.A759D207EC@au-mel-rhel-1.example.com>
Jan 10 10:43:29 au-mel-rhel-1 postfix/qmgr[9596]: A759D207EC: ➡
from=<ataylor@example.com>, size=492, nrcpt=1 (queue active)
Jan 10 10:43:29 au-mel-rhel-1 postfix/smtpd[9599]: disconnect from ➡
unknown[192.168.0.128]
Jan 10 10:43:29 au-mel-rhel-1 spamd[5649]: spamd: connection from ➡
localhost [127.0.0.1] at port 51641
Jan 10 10:43:29 au-mel-rhel-1 spamd[5649]: spamd: setuid to mail succeeded
Jan 10 10:43:29 au-mel-rhel-1 spamd[5649]: spamd: processing message ➡
<20090109234329.A759D207EC@au-mel-rhel-1.example.com> for ➡
mail:8
Jan 10 10:43:29 au-mel-rhel-1 spamd[5649]: spamd: identified spam (1003.6/5.0) ➡
for mail:8 in 1.6 seconds, 1304 bytes.
Jan 10 10:43:29 au-mel-rhel-1 spamd[5649]: spamd: result: Y 1003 - ➡
BAYES_05,GTUBE,MISSING_MID,scantime=1.6,size=1304,user=mail,➡
uid=8,required_score=5.0,rhost=localhost,raddr=127.0.0.1,➡
mid=<20090109234329.A759D207EC@au-mel-rhel-1.example.com>,➡
bayes=0.011644,autolearn=spam
```



```
Jan 10 10:43:29 au-mel-rhel-1 spamd[5363]: prefork: child states: II
Jan 10 10:43:29 au-mel-rhel-1 postfix/pickup[9594]: CCDA7207F2: uid=8 ➡
from=<ataylor@example.com>
Jan 10 10:43:29 au-mel-rhel-1 postfix/cleanup[9608]: CCDA7207F2: ➡
message-id=<20090109234329.A759D207EC@au-mel-rhel-1.example.com>
Jan 10 10:43:29 au-mel-rhel-1 postfix/pipe[9609]: A759D207EC: ➡
to=<jsmith@example.com>, relay=spamassassin, delay=0.17, delays=0.03/0.01/0/0.12, ➡
dsn=2.0.0, status=sent (delivered via spamassassin service)
Jan 10 10:43:29 au-mel-rhel-1 postfix/qmgr[9596]: A759D207EC: removed
Jan 10 10:43:29 au-mel-rhel-1 postfix/qmgr[9596]: CCDA7207F2: ➡
from=<ataylor@example.com>, size=812, nrcpt=1 (queue active)
Jan 10 10:43:29 au-mel-rhel-1 postfix/local[9613]: CCDA7207F2: ➡
to=<jsmith@example.com>, relay=local, delay=0.11, delays=0.06/0.01/0/0.03, ➡
dsn=2.0.0, status=sent (delivered to maildir)
Jan 10 10:43:29 au-mel-rhel-1 postfix/qmgr[9596]: CCDA7207F2: removed
```

在列表 10-15 中，我们连接到 Postfix 服务器，通过了身份验证，并提交了邮件。在处理期间该邮件被提交给 SpamAssassin（请看带 spamd 的那些行——spamd 是 SpamAssassin 后台程序）。

```
Jan 10 10:43:29 au-mel-rhel-1 spamd[5649]: spamd: connection from ➡
localhost [127.0.0.1] at port 51641
Jan 10 10:43:29 au-mel-rhel-1 spamd[5649]: spamd: setuid to mail succeeded
Jan 10 10:43:29 au-mel-rhel-1 spamd[5649]: spamd: processing message ➡
<20090109234329.A759D207EC@au-mel-rhel-1.example.com> for mail:8
Jan 10 10:43:29 au-mel-rhel-1 spamd[5649]: spamd: identified spam ➡
(1003.6/5.0) for mail:8 in 1.6 seconds, 1304 bytes.
Jan 10 10:43:29 au-mel-rhel-1 spamd[5649]: spamd: result: Y 1003 ➡
BAYES_05,GTUBE,MISSING_MID,scantime=1.6,size=1304,user=mail,uid=8,➡
required_score=5.0,rhost=localhost,raddr=127.0.0.1,➡
mid=<20090109234329.A759D207EC@au-mel-rhel-1.example.com>,➡
bayes=0.011644,autolearn=spam
```

我们可以看到，SpamAssassin 后台程序获得了来自服务器的一个连接，把用户改为 mail，然后处理邮件信息。它返回了一行，表示这条信息是一个分数为 1003.6 的垃圾邮件（超出 5.0 非常多，因此是垃圾邮件）。我们可以看到，最后一行的信息 autolearn=spam 表明 SpamAssassin 还从该邮件中学习。

这意味着 Bayesian 信息数据库会取这封邮件中的特征，并把它认作垃圾邮件。当以后的邮件进来时，这个信息会被用来估量邮件是垃圾邮件还是非垃圾邮件（ham）。

我们还可以查看实际的邮件，看看 SpamAssassin 添加的报头资料。

```
$ cat /home/jsmith/Maildir/new/1231544609.V801I22129M888361.mail.example.com
Return-Path: <ataylor@example.com>
X-Original-To: jsmith@example.com
Delivered-To: jsmith@example.com
Received: by au-mel-rhel-1.example.com (Postfix, from userid 8)
id CCDA7207F2; Sat, 10 Jan 2009 10:43:29 +1100 (EST)
X-Spam-Checker-Version: SpamAssassin 3.2.5 (2008-06-10) on
au-mel-rhel-1.localdomain
X-Spam-Level: *****
X-Spam-Flag: YES
X-Spam-Status: Yes, score=1003.6 required=5.0 tests=
```



```
BAYES_05,GTUBE,MISSING_MID,AWL autolearn=spam
version=3.2.5
Received: from unknown (unknown [192.168.0.128])
by au-mel-rhel-1.example.com (Postfix) with ESMTPSA id A759D207EC
for <jsmith@example.com>; Sat, 10 Jan 2009 10:43:29 +1100 (EST)
Date: Sat, 10 Jan 2009 10:43:25 +1100
To: jsmith@example.com
From: ataylor@example.com
Subject: Test spam mail (GTUBE)
X-Mailer: swaks v20061116.0 jetmore.org/john/code/#swaks
Message-Id: <20090109234329.A759D207EC@au-mel-rhel-1.example.com>
```

可见，SpamAssassin 添加了 4 个报头。

- **X-Spam-Checker-Version:** SpamAssassin 的版本。
- **X-Spam-Level:** 用星号表示的总得分。
- **X-Spam-Flag:** 只有邮件被识别为垃圾邮件才会出现。
- **X-Spam-Status:** 垃圾邮件状态 (No 或 Yes)、该邮件获得的总分以及所经过的垃圾邮件测试清单。

5. 获得 SpamAssassin 相关的帮助

与 Postfix 一样，获得 SpamAssassin 的相关帮助很容易。大量的在线文档和广泛而有益的社区都可利用。寻求帮助的最好起点是 SpamAssassin 主页 <http://spamassassin.apache.org/>。该主页包括一个 wiki、一个 FAQ 和一个文档。你还可以到 <http://wiki.apache.org/spamassassin/Mailinglists> 加入非常活跃的邮寄列表，而且可以在 <https://issues.apache.org/SpamAssassin/index.cgi> 上提交程序错误。

至于有关 Postfix 和垃圾邮件的更广泛讨论，可阅读 <http://www.postfix.org/uce.html>。

记住，如果要提交问题或程序错误，应该包含下面的信息。

- SpamAssassin 和 Perl 版本（运行带 `--version` 选项的 `spamassassin` 命令）。
- 平台（运行 `uname -a`）。
- 所产生的全部日志消息（或者在 Ubuntu 的 `/var/log/mail.log` 文件里，或者在 Red Hat 的 `/var/log/maillog` 文件里）。

另外一个可利用的方式是 Freenode IRC 服务器 (<http://freenode.net/>) 上叫作 `#spamassassin` 的 IRC 通道，你可以在那里寻求帮助。

6. 如何处理垃圾邮件？

如果电子邮件被 SpamAssassin 识别为垃圾邮件，那么怎样处理它？在大部分情况下，人们把垃圾邮件放进一个单独的文件夹，复查之后通常会删除它。有些人丢弃或删除所有被标记为垃圾邮件的邮件，但是对此一定要很谨慎。垃圾邮件检测工具（包括 SpamAssassin）不总是正确的，有时会产生错误的决定。这意味着一个合法的电子邮件可能被标记为垃圾邮件。如果删除垃圾邮件，就会失去这封邮件。把它存储在一个文件夹中一段时间，就有可能发现并重新找到误判的邮件。

我们将使用第一种方法，把垃圾邮件放进 Maildir 里一个特别的文件夹以备后查。为了做

这件事，我们可以用两个主要的方法来利用新获得的报头，以便把邮件移动到希望它去的地方。这些方法如下所述。

- 如 procmail 或 maildrop 这样的 MDA。
- 邮件客户端的规则或过滤功能。

使用 procmail 和 maildrop

本章前面的补充知识“邮件投递代理和邮件筛选”中谈到过如 procmail 和 maildrop 这样的 MDA。MDA 是位于 MTA 和邮箱之间的电子邮件筛选引擎。使用这些 MDA 可以创建处理邮件的规则，这些规则基于某些特征，如邮件发件人或它所包含的某个报头。这些规则可以设置为全局适用，或者基于每个用户进行设置。

MDA 的选择完全是个人偏好。许多人喜欢 procmail，另外一些人更喜欢 maildrop。我们不推荐任何偏好，建议两个都尝试一下，然后找到更适合的那一个。

■提示：还有另外一个叫作 Sieve 的 MDA，它经常与 Dovecot 的 IMAP 和 POP 服务器一起使用。你可以在 <http://wiki.dovecot.org/LDA/Sieve> 上阅读 Sieve 的相关内容。许多人发现它比 procmail 和 maildrop 更易用，但是它没那么有名，支持和文档也较少。

本节将说明如何利用这两个工具把垃圾邮件移动到一个适当的文件夹。我们从安装 procmail 和 maildrop 开始。在 Red Hat 上我们发出下面的命令。

```
$ sudo yum install procmail
```

Red Hat 上没有 maildrop 软件包，但可以使用第三方 RPM（如 ftp.silfreed.net/repo/rhel/5/i386/silfreednet/RPMS/maildrop-2.0.4-1.el5.i386.rpm）。我们还可以按照第 7 章所讲的，从源代码（见 <http://www.courier-mta.org/rpm.html>）构建一个 RPM。

或者在 Ubuntu 上，我们安装以下两个软件包。

```
$ sudo aptitude install maildrop procmail
```

现在，我们创建 Spam 文件夹；本例中将为 jsmith 用户创建。在 Red Hat 和 Ubuntu 上，我们可以使用来自 maildrop 软件包的 maildirmake 命令。

```
$ sudo maildirmake -f Spam /home/jsmith/Maildir
```

或者我们使用手动操作更多的 mkdir 命令。

```
$ sudo mkdir /home/jsmith/Maildir/.Spam/{cur,new,tmp};  
chmod -R 0700 /etc/skel/Maildir/.Spam
```

这里创建了 Spam 文件夹并给了它适当的权限。

■提示：我们还可以把这个文件夹添加到/etc/skel 目录，以确保为每个新创建的用户添加该文件。

现在需要告诉 Postfix 使用 MDA 发送电子邮件。为此，我们必须更新 main.cf 文件中的 mailbox_command 配置选项。记得早前介绍这个命令时把它设置为空，从而关闭了对 MDA 的使用。

(1) Procmail

为了使用 procmail，我们如这里演示的这样更新 mailbox_command。

```
mailbox_command = /usr/bin/procmail -a "$EXTENSION" ➡
"DEFAULT=$HOME/Maildir/" "MAILDIR=$HOME/Maildir/"
```

它告诉 Postfix 使用 procmail 命令投递电子邮件。我们添加了一些选项告诉 procmail 查找 Maildirs 的地方。如果没有这些选项，procmail 会试图把邮件投递给一个 mbox 文件。

■注：你必须拥有 root 用户的别名才能使用 procmail。别名必须把 root 用户的邮件发送给真实的用户，正如补充知识“别名”中所讲的那样。如果没有这样的别名，procmail 会失效。

重新载入 Postfix 后，你可以立即尝试给 Postfix 投递一封电子邮件。然后你就可以在系统日志文件里看到一个输入项，表示 procmail 投递了这封电子邮件。

```
Jan 11 13:07:40 au-mel-rhel-1 postfix/local[24520]: 899E5A1235: ➡
to=<jsmith@example.com>, relay=local, delay=0.07, delays=0.04/0.01/0/0.03, ➡
dsn=2.0.0, status=sent (delivered to command: /usr/bin/procmail ➡
-a "$EXTENSION" "DEFAULT=$HOME/Maildir/" "MAILDIR=$HOME/Maildir/")
```

当处理电子邮件时，procmail 寻找有关如何处理这封邮件的指令，也称作处方 (recipe)。它在两个文件寻找这些处方：/etc/procmailrc 和用户主目录里名为 .procmailrc 的文件。/etc/procmailrc 文件包含应用于所有用户的全局指令，而 .procmailrc 文件允许个别用户为他们自己配置专门的指令。

现在我们往全局性的/etc/procmailrc 文件添加 procmail 处方，如列表 10-16 所示。

列表 10-16 处理垃圾邮件的 procmail 处方

```
:0:
* ^X-Spam-Status: Yes
.Spam/
```

这个十分简单的脚本检查所有进来的邮件，查找一个包含 Yes 的被称作 X-Spam-Status 的报头，并把所有匹配的电子邮件移到.spam/文件夹。procmail 命令知道如何找到这个文件夹，因为 Postfix 的 main.cf 文件告诉它用户邮箱包含在它们的主目录中一个叫作 Maildir 的目录里。末尾的/很重要，因为它告诉 procmail 正在往 Maildir 投递邮件。

■提示：如果更喜欢把这个处方限于选定的用户，那么你可以把它放到用户主目录下单独的 .procmailrc 文件里（并考虑把这样一个文件包含在/etc/skel 目录里以便在创建新用户时添加它），而不是全局设置它。

(2) Maildrop

为了改用 maildrop 命令，我们把 mailbox_command 选项设置为以下内容。

```
mailbox_command = /usr/bin/maildrop -d ${USER}
```

该命令告诉 Postfix 给 maildrop 命令投递电子邮件，-d 选项告诉 maildrop 在一个叫作投递模式的特别模式下运行。在投递模式中，maildrop 命令以 \${USER} 所指定的用户身份运

行，通常这个用户就是正在为其投递邮件的那个用户。在此模式下，`maildrop` 命令只有正在写入其邮箱的那个用户的权限，这为防止意外或恶意使用 `maildrop` 命令提供了安全保障。

我们还需要往 `main.cf` 配置文件添加另外一个选项。

```
local_destination_concurrency_limit=1
```

它限制 `maildrop` 一次只投递一封邮件。这避免了 `maildrop` 试图同时投递多封电子邮件时的问题。

注：与 `procmail` 一样，你必须有 `root` 用户的别名才能使用 `maildrop`。别名必须把 `root` 用户的邮件发送给真正的用户，正如补充知识“别名”中所讲的那样。如果没有这样的别名，`maildrop` 会失效。

重新载入或重启 `Postfix` 之后，为了加载新的配置，你可以继续配置 `maildrop`。

同样类似于 `procmail`，你需要告诉 `maildrop` 命令正在写入一个 `Maildir`。为此，我们创建一个叫作 `/etc/maildroprc` 的全局 `maildrop` 处方文件，并告诉 `maildrop` 要写入一个 `Maildir`。我们创建 `/etc/maildroprc` 文件并给它添加下面的内容。

```
$ sudo vim /etc/maildroprc
DEFAULT="$HOME/Maildir"
```

注：在某些发行版上，此文件可能在安装软件包时就添加了，因此它可能已经存在。

这个选项告诉 `maildrop` 可查找用户邮箱的地方，并且现在可以尝试投递一封邮件。如列表 10-17 所示，`Postfix` 系统日志显示 `maildrop` 命令投递了邮件。

列表 10-17 使用 `maildrop` 命令的 `Postfix`

```
Jan 11 16:06:22 au-mel-rhel-1 postfix/local[15314]: DCD56207FC: ➡
to=<jsmith@example.com>, relay=local, delay=0.11, delays=0.06/0.01/0/0.03, ➡
dsn=2.0.0, status=sent (delivered to command: /usr/bin/maildrop -d ${USER})
```

既然 `maildrop` 正在投递邮件，那么你可以配置它把垃圾邮件发送到适当的文件夹。为此，我们往 `/etc/maildroprc` 文件添加一个处方，如列表 10-18 所示。

提示：你也可以在用户的主目录里叫作 `.mailfilter`（在 `maildrop` 里与 `.procmailrc` 文件对等的文件）的文件中指定这个处方。这样你就对哪些用户把邮件移动到那个文件夹潜在的更有选择性。

列表 10-18 处理垃圾邮件的 `maildrop` 处方

```
if ((/^X-Spam-Status:.Yes/))
{
to "./Maildir/.Spam/"
}
```

列表 10-18 只是测试 `X-Spam-Status` 报头是否被设置为 `Yes`，如果是 `Yes`，就把那封邮件发送到 `Maildir/.Spam` 文件夹。

■注：这仅仅抓住了 procmail 和 maildrop 功能的一点皮毛。你非常值得去阅读这两个工具的更多信息，了解它们的其他功能。

在 <http://www.ii.com/internet/robots/procmail/qs/>、<http://lipas.uwasa.fi/~ts/info/proctips.html> 和 <http://pm-doc.sourceforge.net/pm-tips.html> 上可以找到如何使用 procmail 的说明和教程。或者你可以阅读 procmail 的 man 页。

关于 maildrop 的操作说明和教程可以在 <http://www.courier-mta.org/maildrop/maildropfilter.html> 和 <http://www.courier-mta.org/maildrop/maildroptips.html> 上找到。或者你可以阅读 maildrop 的 man 页。

使用邮件客户端

大部分邮件客户端也能够基于邮件所包含的报头过滤它们。某些客户端更显完善；例如 Evolution 和 KMail 两者都允许用户对进来的电子邮件执行反垃圾邮件的过滤功能。从图 10-7 可以看到 KMail（KDE 中默认的邮件客户端）的过滤界面。

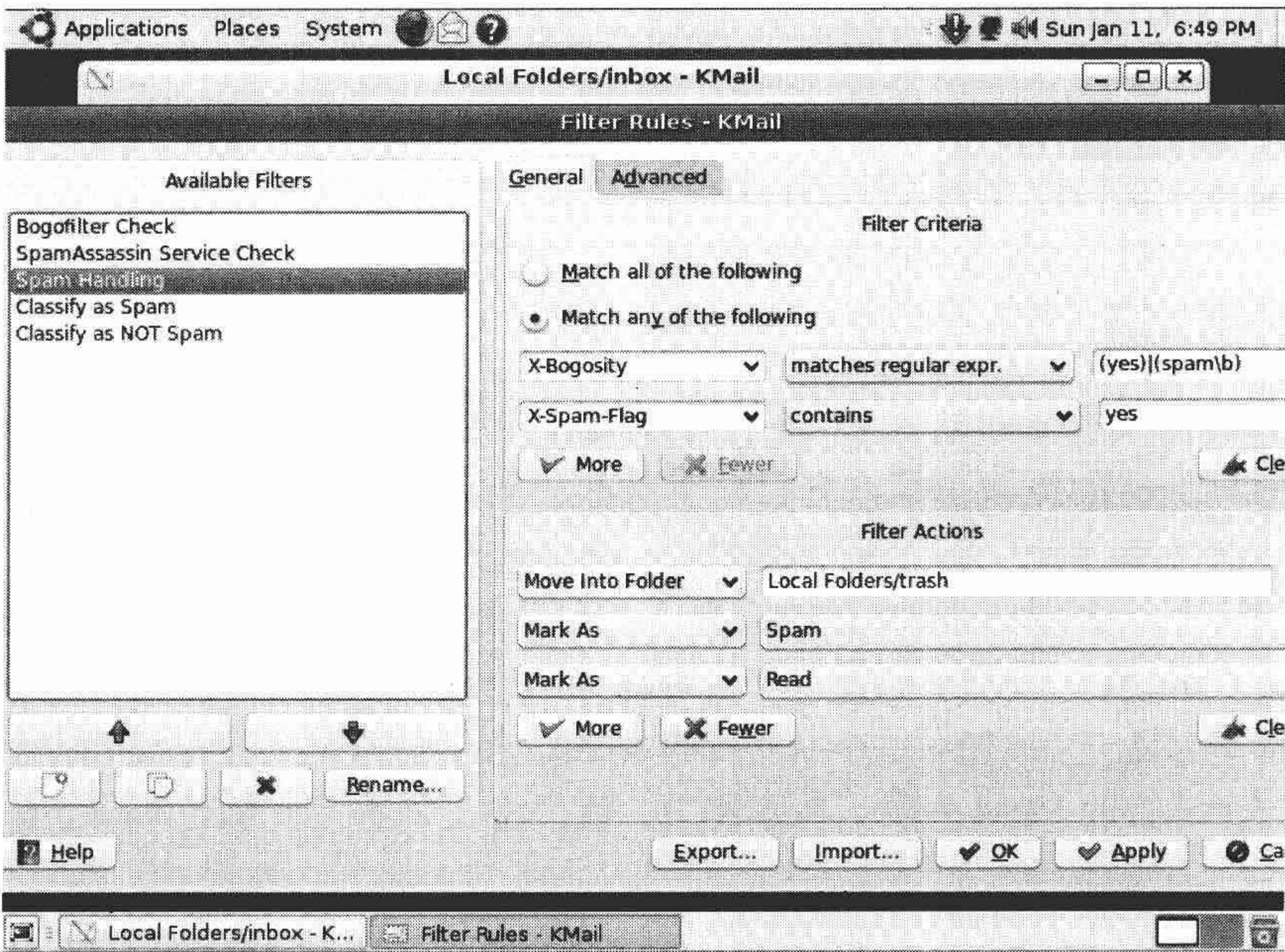


图 10-7 KMail 过滤界面

包含相应规则或过滤系统的其他客户端和界面包括下面这些（除了别的以外）。

- Mozilla Thunderbird。
- Microsoft Outlook。
- Mutt。

10.5.2 防病毒

病毒是为攻击主机、窃取数据或者损害组织其他方面而设计的恶意代码。虽然在 Linux

发行版上极少，但一些病毒把目标瞄向 Linux 发行版。不过，我们并不担心病毒通过电子邮件攻击 Linux 主机；要关心的是病毒可能通过电子邮件传播到其他更易感染的主机，如 Microsoft Windows 台式电脑，或者从你的组织传播到别的组织。

我们将介绍一个叫作 ClamAV 的应用程序，并说明如何把它与刚配置过的 SpamAssassin 装备集成在一起。ClamAV 是一个开源的防病毒引擎，很像 Symantec 和 McAfee 这类公司的类似工具。与这些商业产品不同的是，ClamAV 软件和它的病毒特征更新是免费的。

■注：ClamAV 使用叫作病毒特征（signatures）的特殊规则扫描进来的邮件，寻找看起来像病毒的数据。每个特征包含一个特定病毒的信息，如病毒文件包含的一连串数据，当找到这个信息时，就告诉 ClamAV 检测到一个病毒。

我们将说明如何把 ClamAV 作为一个插件集成到 SpamAssassin 里。SpamAssassin 允许使用插件，通过插件可以添加额外的功能和扫描技术，如病毒扫描。这也是一个无需对配置做重大改动就可扩展 SpamAssassin 的快速又简便的方法。

■注：还有其他更复杂的集成 ClamAV 与 SpamAssassin、Postfix 的方式，其中一些方式可以到 http://www.falkotimme.com/howtos/spamassassin_clamav_procmail/index.php、http://www.section6.net/wiki/index.php/Setting_up_Postfix_Spamassassin_Amavisd_Clamav 和 <http://www.fatofthelan.com/articles/articles.php?pid=22> 了解。

1. 安装 ClamAV

首先，我们需要安装和配置 ClamAV 扫描器和它的后台程序 clamd。我们还需要安装一个叫作 FreshClam 的更新工具，它自动下载和更新病毒特征，ClamAV 利用这些特征检测病毒。

在 Red Hat 上，安装 ClamAV 的过程有点复杂，因为 Red Hat 本身没有装载 ClamAV 产品。为了安装 ClamAV，我们要添加一个由 Dag Wieers 运行的著名的可信任第三方 RPM 库。为此，我们需要安装一个增加对这个库支持的 RPM。在 <http://dag.wieers.com/rpm/FAQ.php#B2> 上可以看到一个适合的 RPM 列表。在本例中，我们将安装适用于 Red Hat Enterprise Linux 5 的 RPM。

```
$ sudo rpm -Uhv http://apt.sw.be/redhat/el5/en/i386/rpmforge/RPMS/
rpmforge-release-0.3.6-1.el5.rf.i386.rpm
```

然后我们可以从这个库安装 ClamAV，而且作为先决条件的一些附加包可能也要安装。

我们还需要安装另外一个包 File::Scan::ClamAV，使用这个包可以使 Clam 作为一个插件集成到 SpamAssassin 中。

```
$ sudo yum install clamav clamd perl-File-Scan-ClamAV
```

在 Ubuntu 上，我们安装 clamav 和 clamav-daemon 软件包。作为先决条件的某些附加包可能也要安装。

```
$ sudo aptitude install clamav clamav-daemon
```


允许 ClamAV 以插件的形式与 SpamAssassin 集成的应用程序 File::Scan::ClamAV 作为一个软件包在 Ubuntu 上不可用。不过，我们可以使用一个叫作 CPAN 的工具（Comprehensive Perl Archive Network）在 Ubuntu 上安装此包。CPAN 是一个有用工具和程序的集合，这些工具和程序是用一个叫作 Perl 的编程语言写的。

为了安装此包，我们启动 CPAN。

```
$ sudo cpan
```

cpan 运行之后，它会提示配置此程序。

```
Would you like me to configure as much as possible automatically? [yes]
```

接受自动配置进程。你可以看到配置进程的执行，并最终呈现一个命令提示符。

```
cpan[1]>
```

在这个命令行上，我们键入下面的命令来安装 File::Scan::ClamAV 软件包。

```
cpan[1]> install File::Scan::ClamAV
```

这里使用了 install 命令和软件包的名字，并用一个叫作 notest 的选项对它预先做了处理。它可能会提示连接到 Internet，如果是这样，回答 yes 继续。然后 File::Scan::ClamAV 软件包就会被安装了。

所有相关软件包安装完成后，你需要确保 ClamAV 后台程序已经启动。在 Red Hat 上，我们启动 clamd 服务。

```
$ sudo service clamd start
```

■注：在 Red Hat 上，当运行 SELinux 时，ClamAV 不能正确启动。有两个选项可以解决这个问题。第一个是用命令 `setsebool -P clamd_disable_trans=1` 为 ClamAV 关闭 SELinux。或者你可以使用 audit 命令更新 SELinux 策略。

在 Ubuntu 上，我们开启 clamav-daemon 服务和 clamav-freshclam 服务。

```
$ sudo invoke-rc.d clamav-daemon start
```

```
$ sudo invoke-rc.d clamav-freshclam start
```

2. 配置 ClamAV

一般不需要修改 ClamAV 的配置选项，但是你应对它的配置方式了解一点。在 Red Hat 上，ClamAV 后台程序通过 `/etc/clamd.conf` 文件配置。clamd.conf 文件配置 ClamAV 后台程序。软件包安装进程还为 FreshClam 更新工具创建了一个每天更新一次 ClamAV 特征的定时任务输入项（在 `/etc/cron.daily` 目录里）。

在 Ubuntu 上，ClamAV 后台程序通过 `/etc/clamav/clamd.conf` 配置文件配置，freshclam 更新后台程序通过 `/etc/clamav/freshclam.conf` 配置文件配置。FreshClam 服务在 Ubuntu 上不是一个定时任务，而是作为一个后台程序运行，它会每天数次下载任何可用的特征。

在 Red Hat 和 Ubuntu 上，运行 ClamAV 会创建一个 Unix 套接字文件。这个特殊类型的文件被 SpamAssassin 用来与防病毒扫描器通信。电子邮件被提交给这个套接字，被扫描，然

后带着是否是病毒的评估结果返回给 SpamAssassin。接下来这个评估结果以 X-Spam-Virus 报头的形式添加到扫描过的电子邮件中。

在 Red Hat 上，这个套接字默认位于/tmp/clamd.socket，在这里你可以看到一个有关它的列表。

```
$ ls -la /tmp/clamd.socket
srwxrwxrwx 1 clamav clamav 0 Jan 12 21:18 /tmp/clamd.socket
```

在 Ubuntu 上，这个套接字位于/var/run/clamav 目录，被称为 clamdctl。

注：要知道这个套接字的位置，这样就可以告诉 SpamAssassin 何时提交电子邮件以备扫描。你可以在 clamd.conf 配置文件里调整这个套接字文件的位置。

3. 配置 SpamAssassin

当运行 ClamAV 时，你需要把它集成到 SpamAssassin。为此，你需要创建一个插件。我们打算把插件存储在目录/etc/mail/spamassassin。此目录将由两个文件组成：clamav.cf 和 clamav.pm。clamav.cf 文件是一个 SpamAssassin 配置文件，clamav.pm 文件包含 ClamAV 插件所必需的代码。

从列表 10-19 可以看到 clamav.cf 文件的内容。

列表 10-19 clamav.cf 文件

```
loadplugin ClamAV clamav.pm
full CLAMAV eval:check_clamav()
describe CLAMAV Clam AntiVirus detected a virus
score CLAMAV 10
```

在列表 10-19 中，clamav.cf 文件配置 ClamAV 插件。它加载这个插件并描述用于检测病毒的特殊规则。当这个规则被触发时（例如，当检测到一个病毒时），被感染的电子邮件就加 10 分。这个分数超过默认的 5.0 分，因此它自动认为检测到的病毒都是垃圾邮件。

clamav.cf 文件所加载的 lamav-pm 文件是一个 SpamAssassin 插件。从列表 10-20 可以看到该插件的内容。

列表 10-20 clamav.pm 插件

```
package ClamAV;
use strict;

our $CLAMD_SOCKET = "/var/run/clamav/clamdctl"; # change me
#our $CLAMD_SOCKET = "/tmp/clamd.sock"; # change me
#our $CLAMD_SOCKET = 3310; # for TCP-based usage

use Mail::SpamAssassin;
use Mail::SpamAssassin::Plugin;
use File::Scan::ClamAV;
our @ISA = qw(Mail::SpamAssassin::Plugin);

sub new {
    my ($class, $mailsa) = @_;
    $class = ref($class) || $class;
```



```

my $self = $class->SUPER::new($mailsa);
bless ($self, $class);
$self->register_eval_rule ("check_clamav");
return $self;
}

sub check_clamav {
    my ($self, $permstatus, $fulltext) = @_;
    my $clamav = new File::Scan::ClamAV(port => $CLAMD_SOCKET);
    my ($code, $virus) = $clamav->streamscan($fulltext);
    my $isspam = 0;
    my $header = "";
    if(!$code) {
        my $errstr = $clamav->errstr();
        Mail::SpamAssassin::Plugin::dbg("ClamAV: Error scanning: $errstr");
        $header = "Error ($errstr)";
    } elsif($code eq 'OK') {
        Mail::SpamAssassin::Plugin::dbg("ClamAV: No virus detected");
        $header = "No";
    } elsif($code eq 'FOUND') {
        Mail::SpamAssassin::Plugin::dbg("ClamAV: Detected virus: $virus");
        $header = "Yes ($virus)";
        $isspam = 1;
    } else {
        Mail::SpamAssassin::Plugin::dbg("ClamAV: Error, unknown return code: $code");
        $header = "Error (Unknown return code from ClamAV: $code)";
    }
    $permstatus->{main}->{conf}->{headers_spam}->{"Virus"} = $header;
    $permstatus->{main}->{conf}->{headers_ham}->{"Virus"} = $header;
    # add a metadatum so that rules can match against the result too
    $permstatus->{msg}->put_metadata('X-Spam-Virus',$header);
    return $isspam;
}

1;

```

■注: clamav.cf 和 clamav.pm 文件还包含在本书附带的源代码里, 你可以从 Apress 站点下载。你还可以从下面的网站复制粘贴它们: <http://wiki.apache.org/spamassassin/ClamAVPlugin>。

clamav.pm 文件是一个用 Perl 编程语言写的 SpamAssassin 插件。对大部分内容都无需过多担心, 你只需更改一行。

```
our $CLAMD_SOCKET = "/tmp/clamd.sock"; # change me
```

这里显示了 ClamAV 后台程序套接字的位置: 在 Red Hat 上的 /tmp/clamd.sock 或者 Ubuntu 上的 /var/run/clamav/clamdctl。你需要去掉这一行的注释符号, 并确保只去掉以 our \$CLAMD_SOCKET 开头的这一行的注释符号。然后我们重启 SpamAssassin 以开启此插件, 在 Red Hat 上如下所示。

```
$ sudo service spamd restart
```

4. 测试带 ClamAV 的 SpamAssassin

既然开启了 ClamAV 插件, 我们就要测试进来的邮件是否进行了病毒扫描。为了做此事, 我们再次使用 swaks 命令, 演示如下。


```
$ swaks -tls -a -au ataylor -ap password -t jsmith@example.com -f ataylor@example.com
```

这里再次使用 `swaks` 命令给 `jsmith@example.com` 发送了一封电子邮件。

■注：你要将其替换为你平台上适当的用户来测试它。

这封电子邮件会被 MTA 接收和处理，并基于相应配置传递给 SpamAssassin 以做分析。这封邮件将被提交到用户 `jsmith` 的邮箱，然后你可以分析邮件报头的内容。从列表 10-21 可以看到刚发送的这封邮件的报头。

列表 10-21 带 ClamAV 的 SpamAssassin 扫描之后的电子邮件报头

```
Return-Path: <ataylor@example.com>
X-Original-To: jsmith@example.com
Delivered-To: jsmith@example.com
Received: by ubuntu-server.example.com (Postfix, from userid 8)
id 531F820814; Mon, 12 Jan 2009 23:40:21 +1100 (EST)
X-Spam-Virus: No
X-Spam-Checker-Version: SpamAssassin 3.2.5 (2008-06-10) on
au-mel-rhel-1.localdomain
X-Spam-Level:
X-Spam-Status: No, score=-1.4 required=5.0 tests=ALL_TRUSTED,AWL
autolearn=ham, version=3.2.5
```

从列表 10-21 可以看到，一个叫作 `X-Spam-Virus` 的报头已经被添加到这封邮件中。在这里该报头的值是 `No`。如果检测到病毒，这个报头将会被标记为 `Yes`。

5. 如何处理受感染的电子邮件？

与 `X-Spam-Status` 报头一样，这个报头可以用来对被识别为包含病毒的电子邮件做特别处理，例如，把所有这样的电子邮件移到一个叫作 `Viruses` 的隔离文件夹里。这可以使用前面用来把垃圾邮件分类放进单独文件夹的 `procmail` 或 `maildrop` 命令来进行。为了使用 `procmail` 命令做这件事，我们往全局 `/etc/procmailrc` 文件或用户的 `.procmailrc` 文件添加一个如下面所示的处方。

```
:0:
* ^X-Spam-Virus: Yes
.Viruses/
```

要用 `maildrop` 做同样的事情，我们可往全局 `/etc/maildroprc` 文件或用户的 `.mailfilter` 文件添加下面的处方。

```
if ((/^X-Spam-Virus:.*Yes/))
{
to "./Maildir/.Viruses/."
}
```

6. 获得 ClamAV 相关的帮助

你可以找到很多有关 ClamAV 故障检修的资源。我们应该从 ClamAV 的主页开始 (<http://www.clamav.net/>)。在该站点上，你可以找到许多种可用的配套资源 (<http://www.clamav.net/>)。

support/), 而且可以加入 ClamAV 的邮寄列表 (<http://www.clamav.net/support/ml/>)。在 <http://www.clamav.net/bugs/> 上可以找到当前的程序错误列表。

记住, 如果要提交问题或程序错误, 应该包含下面的信息。

- ClamAV 版本 (运行带 `--version` 选项的 `clamscan` 命令)。
- 平台 (运行 `uname -a`)。
- 所产生的全部日志消息 (在 Ubuntu 上 `/var/log/clamav/clamav.log` 或者 `/var/log/clamav/freshclam.log` 文件里, 或者在 Red Hat 上 `/var/log/clamav/clamav.log` 或者 `/var/log/clamav/freshclam.log` 文件里)。

另外一种可以利用的方式是 Freenode IRC 服务器上叫作 `#clamav` 的 IRC 通道, 你可以在那里寻求帮助。

10.6 配置 IMAP 和 POP3

与在本章前面大部分章节中看到的不一樣, 用户不打算通过命令行直接访问他们的电子邮件。他们希望在本地上通过电子邮件客户端访问它。这就是 IMAP 或 POP3 服务器盛行的原因。这两个协议代表从邮件客户端或 MUA 访问邮箱的两种不同方法。我们将分析这两种方法, 阐述每种方法的正反两方面, 并演示如何在你的平台上配置和实现它们。

10.6.1 IMAP

IMAP 用于从远程客户端 (如台式机或膝上电脑) 访问电子邮箱。客户端连接到 IMAP 服务器就可以阅读、管理和删除邮箱里的任何电子邮件。我们还可以搜索消息、创建和删除文件夹以及完成其他多种管理任务。

10.6.2 POP3

POP3 同样用于从远程主机访问电子邮箱。电子邮件被接收并保存在用户的邮箱, 直到用户检查他们的邮件。当一个用户的电子邮件客户端进行连接时, 所有正在等待的电子邮件就被下载到客户端并从服务器移除。

10.6.3 二者有什么区别?

IMAP 协议的行为很像一个文件服务器。电子邮件停留在服务器上, 可以被阅读、删除和处理。POP3 是一个存储转发机构。每个协议都有优点和缺点。

IMAP 的优点如下。

- 允许用户从多个位置访问电子邮件, 不仅仅是他们的客户端。例如, IMAP 允许网

页邮件的访问。

- 保护意外删除的消息。
- 统一管理电子邮件，使备份和恢复更容易。

POP3 的优点如下。

- 不需要连接到服务器（或者网络）就可访问电子邮件。
- 不使用服务器上的任何存储——所有邮件取回之后都存储在客户端。

IMAP 的缺点如下。

- 要连接到服务器（因此要连接到网络和/或 Internet）才能访问电子邮件。
- 服务器必须有足够的存储容量。

POP3 的缺点如下。

- 邮件只存储在客户端，从别的地方不能访问。
- 在没有充分备份的情况下，如果用户失去、损坏或者重建他们的客户端台式机、膝上电脑等，他们就可能失去邮件。
- 如果用户需要备份电子邮件，每个客户端的备份和恢复策略可能复杂而难以实现。

10.6.4 在 IMAP 和 POP3 之间选择

在绝大多数情形下，我们推荐使用 IMAP。原因是它简单易用。IMAP 允许用户用客户端漫游或者没有客户端；例如 IMAP 服务器允许用户通过他的膝上电脑和如 iPhone 或 Blackberry 之类的 PDA 设备访问他的电子邮件。

用户的电子邮件还在一个中心位置，这样更容易为它做备份（以及在其中某个用户不可避免地删除了一个关键邮件时恢复它）。另外，因为存储设备成本低廉，把电子邮件集中存储在服务器或磁盘阵列对大多数组织都不再是问题。

主要的告诫就是，如果用户不能连接到 IMAP 服务器，他们就不能取回邮件。我们认为 IMAP 的其他优点的利益超过这种情况下的风险。

本节会用例子演示 IMAP 的配置方式。如果对配置 POP3 感兴趣，你可以到 <http://wiki.dovecot.org/POP3Server> 和 <http://wiki.dovecot.org/QuickConfiguration> 查看一些操作说明、提示和告诫。

10.6.5 Dovecot 介绍

我们已经引入了 Dovecot 服务器，因为在我们的例子中就是用它作为 Postfix 的身份验证服务。那么，这是一些好消息。如果你一直按照例子操作至此，那么应该已经安装过、启动过并部分配置过 Dovecot。这意味着现在我们只需要采取有限的步骤就可以让它工作。

本节将说明如何打开 IMAP，特别是叫作 IMAPS 的 IMAP 协议的安全版本（S 意思是安全）。它使用 SSL 加密协议在数据流动于客户端和服务端时保护用户的身份验证信息和用户邮件内容。IMAP 和 IMAPS 之间的另一个主要区别是它们运行所在的 TCP 端口。IMAP 协议运行在端口 143，而 IMAPS 协议运行在端口 993。这个端口号需要在邮件客户端里定义（并

暗示客户端要使用 SSL)。

■提示：POP3 协议运行在 TCP 端口 110，它的对应安全版本 SSL-POP3 运行在 TCP 端口 995。

我们还打算配置 Dovecot 以查找本地 Maildir 邮箱。我们不需要配置身份验证功能，因为已经做过了。为 Postfix 开启的身份验证机制使用 PAM 身份验证服务检查本地用户的用户名和密码，同样的机制在 IMAP 连接中会工作得很好。

1. 配置 Dovecot

正如本章前面所述，Dovecot 服务器使用 `dovecot.conf` 配置文件配置。在 Ubuntu 上，这个文件位于 `/etc/dovecot/dovecot.conf`，在 Red Hat 上，该文件位于 `/etc/dovecot.conf`。让我们从开启 IMAPS 协议和指定所希望使用的 SSL 证书位置开始这个配置过程。为了使这个过程更轻松，我们将重新使用为 Postfix 加密所创建的证书。

首先，我们通过编辑 `protocols` 配置选项并把它从 `none` 改为 `imaps` 来开启 IMAPS，如下所示。

```
protocols = imaps
```

接着我们指定 SSL 证书和密钥文件。为做此事，我们需要反注释并更新 `ssl_cert_file` 和 `ssl_key_file` 选项，并添加证书和密钥的位置。

```
ssl_cert_file = /etc/postfix/tls/mail.example.com.cert
ssl_key_file = /etc/postfix/tls/mail.example.com.key
```

这里要使用的证书和密钥文件与为 Postfix 加密所提供的相同。我们可以按照在那一节所进行的相同的步骤创建 Dovecot 专用证书，或者如果愿意的话，你甚至可以专门为 Dovecot 购买另外的证书，不过我们感觉没有这个必要。

■注：记住，证书连接着主机名，在我们的例子就是 `mail.example.com`。如果在别的主机上运行 Dovecot 服务器，我们应该用运行 Dovecot 的服务器的主机名创建一个新密钥和证书。另外，因为证书绑在主机名上，所以我们必须在邮件客户端指定这个主机名（如 `mail.example.com`）而不是通常所说的其他任何 DNS 名。

接下来我们去掉注释符号并把 `disable_plaintext_auth` 选项改为 `yes`。

```
disable_plaintext_auth = yes
```

如果没有开启 SSL，也没有运行加密的连接，该选项就关闭所有的明文身份验证功能。这保护用户的身份验证凭证免受网络或 Internet 的探查和抓取攻击。

最后我们要使用 `mail_location` 选项指定邮箱位置。

```
mail_location = maildir:~/Maildir
```

它告诉 Dovecot 正在使用位于用户主目录或 `~` 中目录 Maildir 下的 Maildirs。

现在要重启 Dovecot 服务器，然后我们就可以测试，看看是否能做一个连接并取回 Ubuntu 上的邮件，如下所示。

```
$ sudo invoke-rc.d dovecot restart
```


2. 测试 Dovecot

既然配置了 Dovecot，我们就可以打开一个客户端并测试它的访问功能。我们将配置一个 Mozilla Thunderbird 客户端来测试 Dovecot。

注：Mozilla Thunderbird 是由 Mozilla Foundation 发布的广受欢迎的开源邮件客户端，Mozilla Foundation 还开发了 Firefox 浏览器。

为了配置客户端，我们首先要安装它。在 Red Hat 和 Ubuntu 上，必需的软件包都叫 thunderbird，我们应该使用软件包管理工具安装它。

现在，让我们回顾一下为配置客户端需知的信息。

- 服务器名。
- 用户名和密码。

通过菜单选项“Applications”→“Internet”→“Thunderbird Email”启动 Mozilla Thunderbird 邮件客户端，如图 10-8 所示（在 Ubuntu 上，该程序从同样的位置启动，但是被称为 Mozilla Thunderbird Mail/News）。

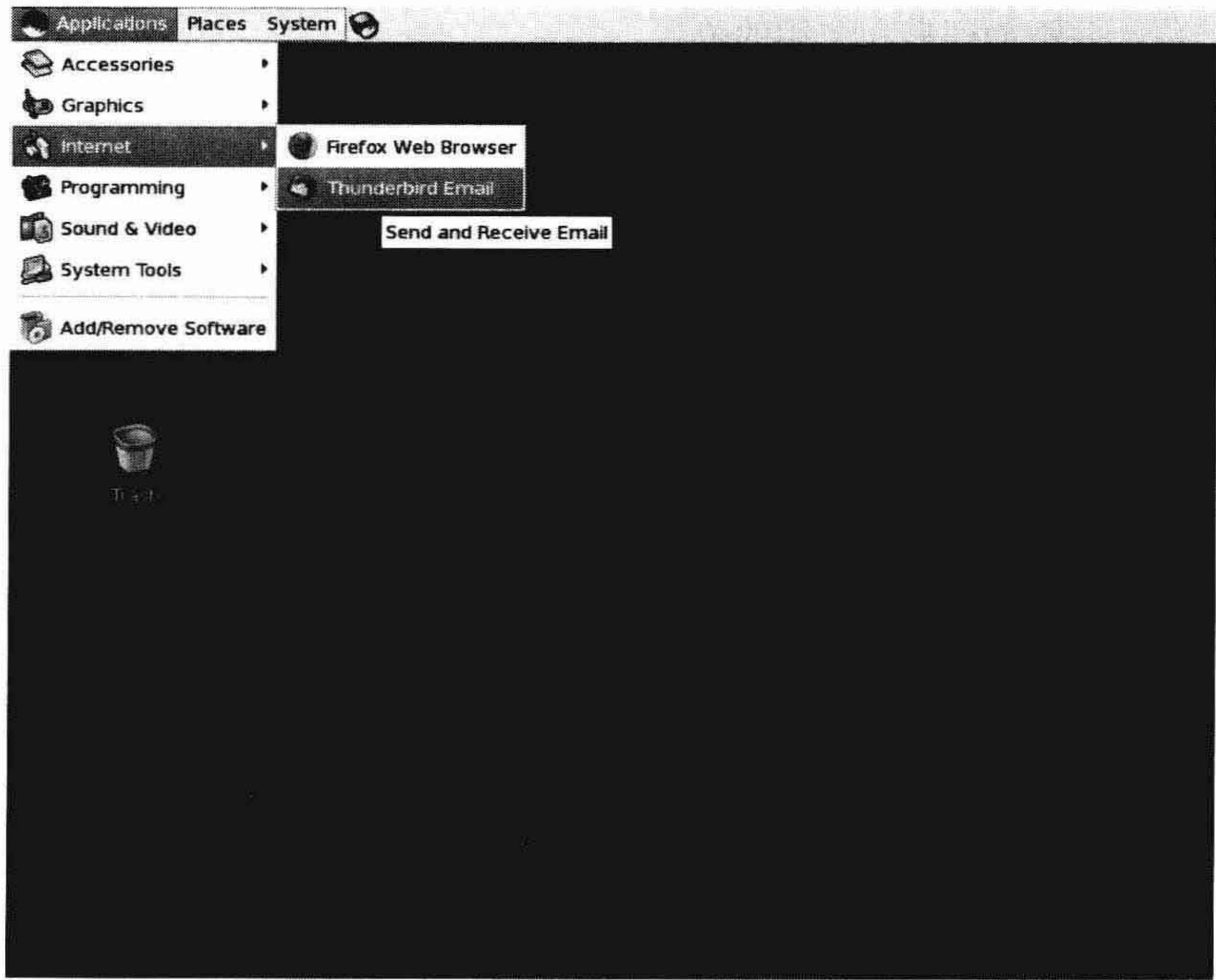


图 10-8 启动 Mozilla Thunderbird

如果是第一次启动 Thunderbird，它会打开账号向导。使用这个向导可以创建多种新型的 Thunderbird 账号。如果此向导没有运行，我们可以通过选择菜单选项“Edit”→“Account Settings”并单击“Add Account”创建一个新账号。

我们打算配置一个新的电子邮件账号。我们确保此选项被选中并单击“Next”按钮。在接下来的一个画面上指定名字和电子邮件地址并再次单击“Next”按钮。

在下面一个画面指定 IMAP 和 SMTP 服务器的类型和名字（Thunderbird 称它为邮件服务器）。从图 10-9 可以看到这一个画面，填写的内容是范例服务器 mail.example.com。

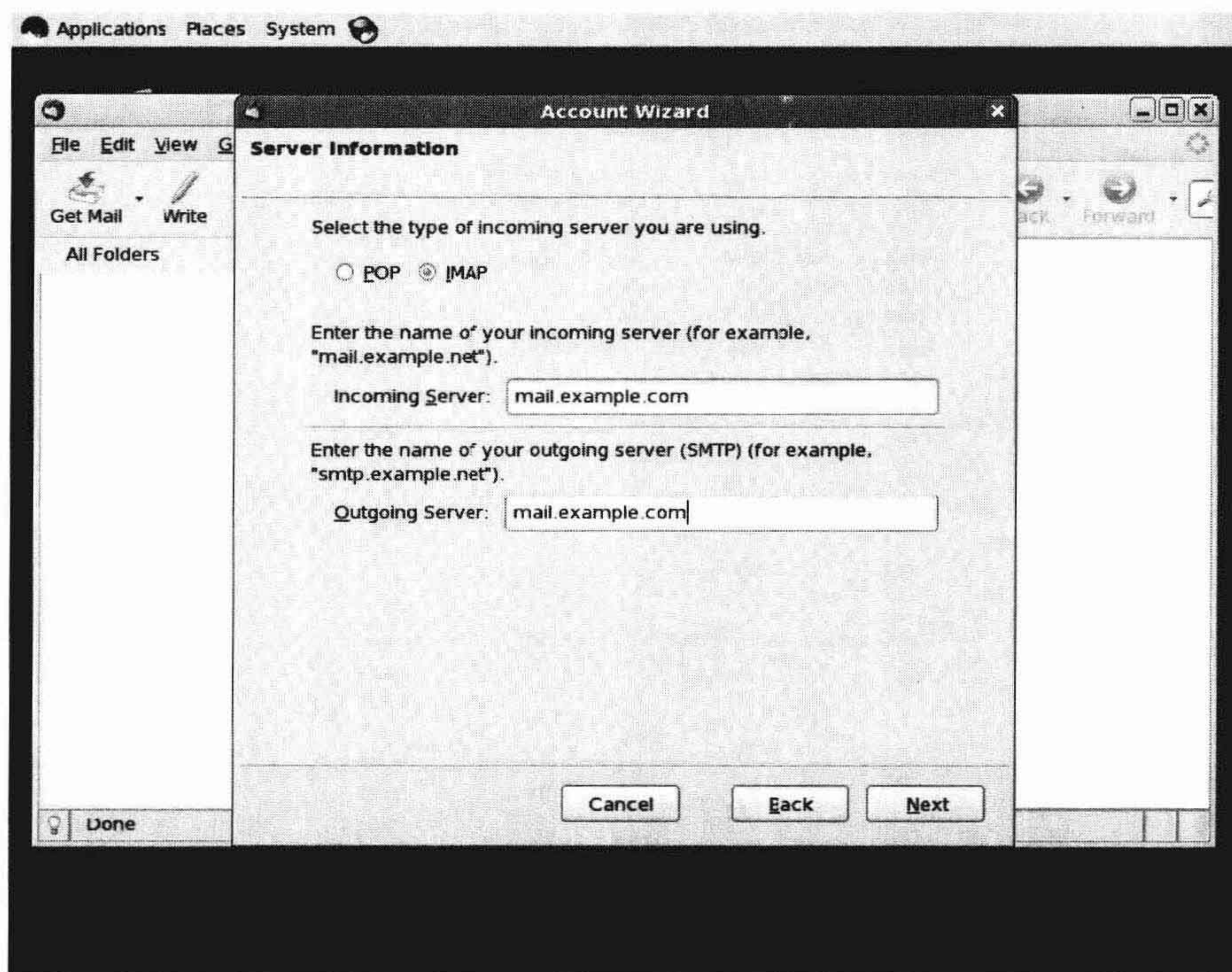


图 10-9 指定服务器资料

指定好服务器资料，单击“Next”按钮。

接下来的一个画面提示输入用户名；在我们的例子里，这是 IMAP 主机上已存在的一个用户名。输入名字后，单击“Next”按钮。

在接下来的两个画面中，指定该账号在 Thunderbird 里的名字。Thunderbird 用这个名字称呼该账号。一般使用默认值，就是我们的电子邮件地址，然后单击“Next”按钮。在最后一个画面，如图 10-10 所示，我们可以复查已输入的资料以确保正确性。

这些资料在本例中是正确的，因此单击“Finish”按钮。如果当到达这一画面时资料不正确，我们可以使用“Back”按钮回去修改资料。

但是这个配置不是十分完整，因为用账号向导配置账号时 Thunderbird 默认设置为 IMAP 而不是 IMAPS。那么完成向导之后，我们需要编辑该账号以便使用 IMAPS。为了做这件事，我们选择“Edit”菜单和“Account Settings”选项启动一个对话框，这里允许编辑账号资料。从图 10-11 可以看到这个对话框。

选择“Server Settings”标签，然后在“Security Settings”框里选中“SSL”按钮。注意端口号被改为 993 表示要做 IMAPS 连接。我们单击“Okay”按钮完成配置。

现在，如果单击“Get Mail”按钮，客户端会通过 TCP 端口 993 连接到 Dovecot 服务器 mail.example.com 并发起一个 SSL 加密的连接。它提示输入密码。当身份验证成功时，邮箱里所有可用的邮件都会显示在 Thunderbird 里，如图 10-12 所示。

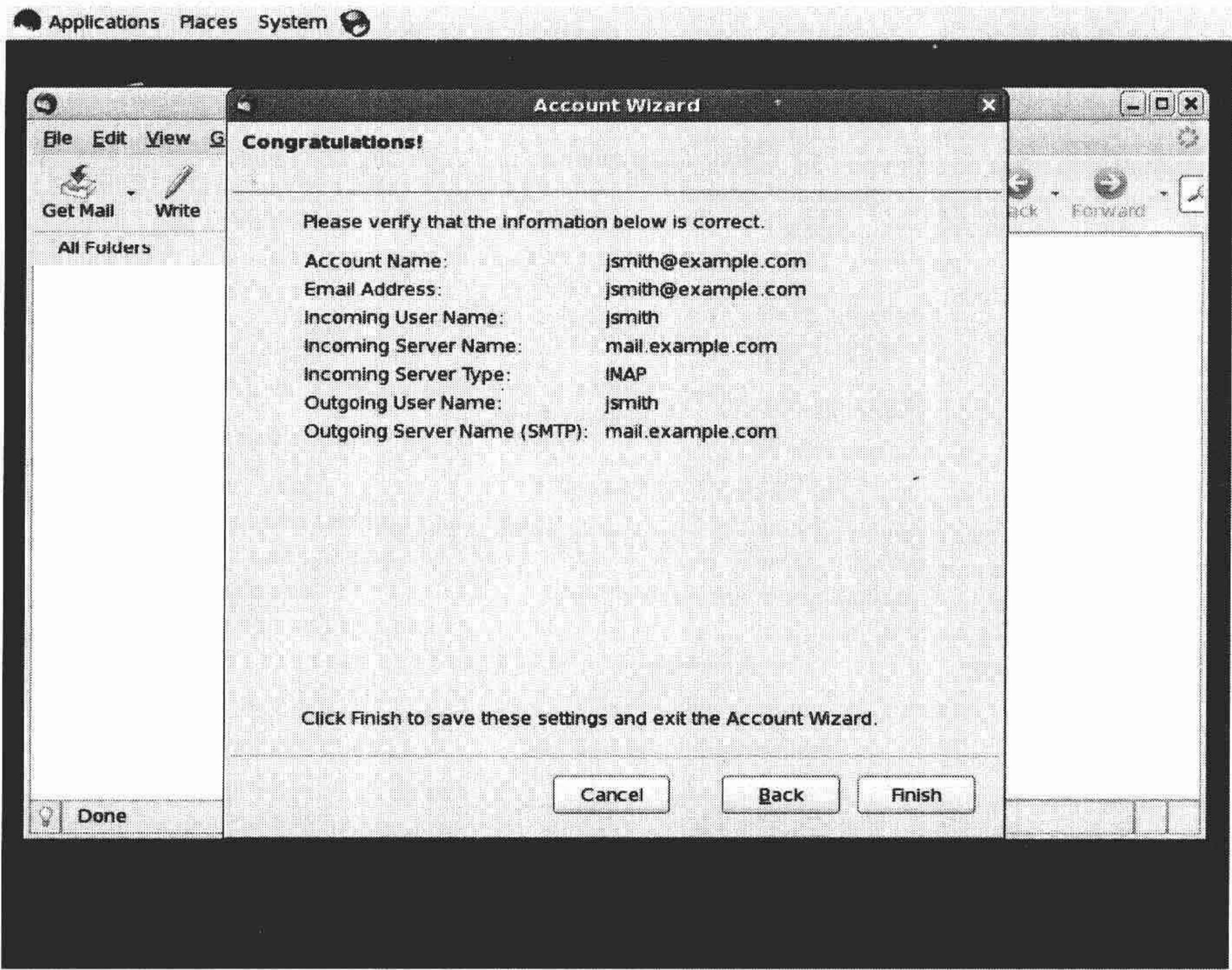


图 10-10 复查账号资料

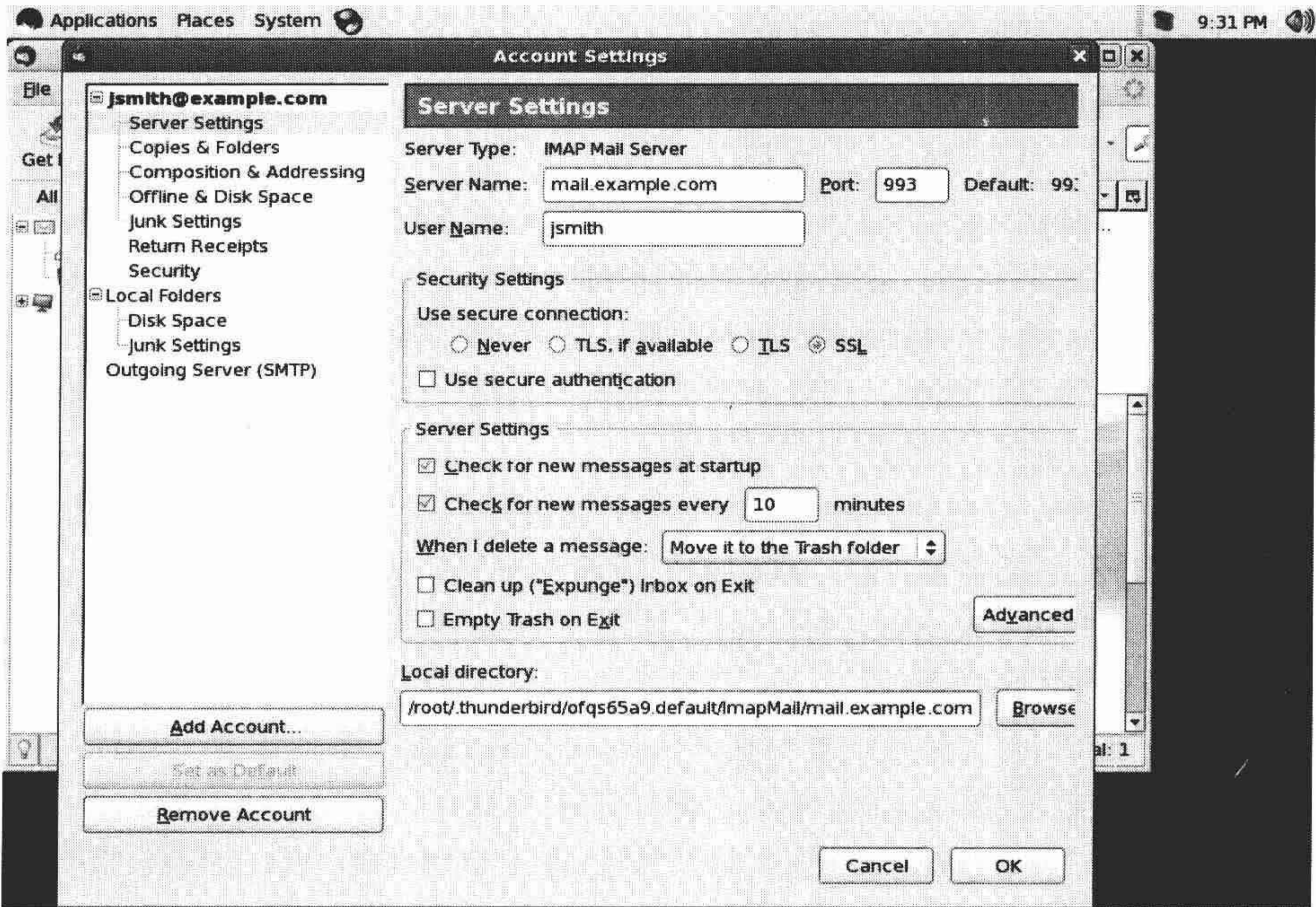


图 10-11 编辑 Thunderbird 账号资料



图 10-12 邮箱

你应该还可以在适当的日志文件里看到连接和身份验证的记录，它们在 Red Hat 上的 /var/log/maillog 里或者在 Ubuntu 上的 /var/log/mail.log 里，如列表 10-22 所示。

列表 10-22 Dovecot 身份验证日志输入项

```
Jan 14 22:13:53 au-mel-rhel-1 dovecot: imap-login: Login: user=<jsmith>, ➡
method=PLAIN, rip=::ffff:192.168.0.128, lip=::ffff:192.168.0.128, TLS
```

3. Dovecot 的故障检修

如果出了故障或者不能连接到 Dovecot 服务器了，那么我们推荐查看的第一个地方是日志文件。检修过程可以更轻松，因为 Dovecot 有一些非常有用的调试设置。如果有问题，你可以打开它们以得到更详细的输出。表 10-3 提供了关于这些选项的清单。

表 10-3 Dovecot 的调试选项

选项	说明
mail_debug	当设置为 yes 时，它显示更多有关邮件处理的信息
auth_verbose	当设置为 yes 时，它显示更多有关身份验证过程的信息
auth_debug	当设置为 yes 时，它显示有关身份验证过程的调试信息
auth_debug_password	如果 auth_debug 和这个选项都设置为 yes，就会显示身份验证机制和密码

打开其中某个或某些选项可看到详细的调试信息。这个信息被发送给系统日志后台程序，并由此转送到日志文件 /var/log/maillog 或 /var/log/mail.log。利用辅助信息可以更容易确定

Dovecot 装备是否存在故障。

■注：你必须在修改这些选项之后重启 Dovecot 服务器，我们推荐在解决问题之后把这些选项设置为之前的 no。继续保持调试状态可能会影响 Dovecot 服务器的性能。

4. 获得 Dovecot 相关的帮助

有数量可观的有用信息可以帮助解决有关 Dovecot 的任何问题。尝试得到帮助的最好的起点是 Dovecot 主页 <http://www.dovecot.org/>。该站点包括一个内容全面的 wiki (<http://wiki.dovecot.org/>)，这个 wiki 包括许多操作指导以及在多种实现中配置 Dovecot 的例子。我们还可以利用的是 Dovecot 邮寄列表 (<http://www.dovecot.org/maillinglists.html>) 和程序错误提交说明 (<http://www.dovecot.org/bugreport.html>)。

记住，如果要提交问题或程序错误，应该包含下面的信息。

- Dovecot 版本（运行带 `--version` 选项的 `dovecot` 命令）。
- 平台（`run uname -a`）。
- 所产生的全部日志信息（在 Ubuntu 的 `/var/log/mail.log` 文件或 Red Hat 的 `/var/log/maillog` 文件里）。

还可以利用的方式是 Freenode IRC 服务器上叫作 `#dovecot` 的 IRC 通道，你可以在那里寻求帮助。

10.7 虚拟域与虚拟用户

本章介绍了为用户提供电子邮件服务的基础知识。第 15 章将介绍一个完整的电子邮件和协作程序组，不过在这里将讲述本章介绍过的邮件服务的可扩展方式。

可以扩展 Postfix 和 Dovecot 的主要方式是扩展到虚拟域和虚拟用户。直到现在，我们一直假定投递邮件所用的域和接收邮件的用户全是实际的实体。接受电子邮件的域是邮件服务器所属的域，而且用户实际存在于邮件服务器上。有了虚拟域和虚拟用户的概念，我们就可以为电子邮件配置非实际的目的地和收件人。我们打算论述这些概念，但是会解释它们并提供一些参考文档，让读者自己去扩展本章所构建的内容并做所有这些事情。

使用虚拟域可以接收其他域的电子邮件。例如，假定我们当前接收域为 `example.com` 的邮件。你的组织可能还有 `product.com` 和 `anotherproduct.com` 这两个域。你可以配置 Postfix 接收这些附加域上的电子邮件。

虚拟用户是一个类似的概念。迄今我们所有的用户都作为实际的 Linux 用户存在，如 `jsmith` 和 `ataylor`。他们的邮箱存储在 `/home` 目录树中。有了虚拟用户，邮件用户就不用创建为操作系统用户。每个电子邮件地址被映射到一个虚拟用户或者如 MySQL 或 LDAP 之类的数据库中的一个用户。这减少了创建和管理大量操作系统用户的需求。

在 http://www.postfix.org/VIRTUAL_README.html 上可以找到一个关于用 Postfix 配置虚拟域和虚拟用户的非常完整的操作指导。在 <https://help.ubuntu.com/community/Postfix>

CompleteVirtualMailSystemHowto 上可以找到一个主要针对 Ubuntu 的指南。你还可以在 <http://wiki.dovecot.org/VirtualUsers> 和 Dovecot 操作指南页面 <http://wiki.dovecot.org/HowTo> 上找到用类似方式扩展 Dovecot 的说明。

10.8 小 结

本章介绍了 Linux 主机上的邮件服务，它允许用户从你的组织收发电子邮件。我们介绍了 IMAP 和 POP3 协议，它们允许用户连接到主机并取回和管理他们邮箱里的邮件。

在这个过程中涵盖了以下一些重要的概念。

- 通过加密保护用户的电子邮件和他们的身份验证凭证。
- 开启 SASL 身份验证功能，以使用户安全地通过邮件服务器的验证以便收发电子邮件。
- 利用 SpamAssassin 过滤电子邮件并阻止垃圾邮件到达用户。
- 使用开源的 ClamAV 防病毒引擎为用户的邮件检查病毒和恶意代码。
- 使用如 procmail、maildrop 或 Sieve 之类的 MDA 过滤电子邮件。
- 实现 Dovecot 服务器，以使用户连接到他们的邮箱并取回电子邮件。

下一章将介绍网络服务，并教你运行自己的网络服务器以及提供自己的网站。

第 11 章 Web 服务和 SQL 服务

在配置好 e-mail 服务后，客户就可以通过 e-mail 联系我们。然而，为了完善网络交流，还需要一个网站，并且还可以为那些需要远程访问 e-mail 的员工架设 webmail，或者架设一个可以方便员工协同工作的 wiki。

本章首先介绍如何架设 Apache 网页服务器和 MySQL 数据库服务器；接着，我们将安装一个内容管理系统和一个 webmail 应用程序；最后还将展示如何通过网络代理来保护我们的工作人员，使其能够享受更快、更安全的上网体验。

11.1 Apache 网页服务器

Apache 可能是如今最为广泛使用的开源软件了。现有 50% 以上的网站¹都在使用它，这些网站选择 Apache 通常是因为它的成熟性、稳定性和灵活性。Apache 被设计成模块化的软件，所以通过启用和禁用相应的模块可以添加和删除它的外部功能。Apache 软件包几乎可用于所有的 Linux 发行版本，因此可使用软件包管理系统安装。

11.1.1 安装和配置

Red Hat 和 Ubuntu 都安装了 Apache2.2 版，但它们的软件包命名不同。在 Red Hat 上应运行“`sudo yum install httpd`”命令；而在 Ubuntu 上则应运行“`sudo aptitude install apache2-pm-prefork`”命令。安装 Apache 软件包还会安装一些附加的库。

读者可能已经注意到在 Ubuntu 上有些可用的 apache2 的附加软件包。这些软件包提供不同的内部引擎，从而允许 Apache 以更为有效的方式为各种各样特殊的 Web 请求提供服务。我们选择 apache2-mpm-prefork 软件包是因为该引擎允许以后为 PHP 脚本语言添加支持。在 Red Hat 上，这些各种各样的引擎都被包含在单独的 httpd 软件包中，可以简单地通过配置文件来启用和禁用它们。为此，还需要在 Red Hat 上安装 system-config-httpd 软件包。

¹ 查看“http://news.netcraft.com/archives/Web_server_survey.html”。

■注：如果运行大容量的、高通信业务量的网站，而且不使用 PHP 脚本，则完全可以考虑换用不同的 Apache 引擎或 Web 服务器，比如 lighthttpd 和 Tux。读者可以在“http://en.wikipedia.org/wiki/Comparison_of_Web_servers”上找到一系列 Web 服务器。

我们将首先为 Apache 做一些基本配置。稍后，再通过添加一些模块以进行功能性的扩展。

1. Red Hat

在 Red Hat 上可通过图形用户界面（GUI）的方式配置 Apache。首先，选择菜单命令“系统>管理>服务器设置>HTTP”以启动配置程序。这里，需要输入 root 用户的密码。该工具提供一个易用的界面，能够在文件/etc/httpd/conf/httpd.conf 中更改 Apache 的基本配置。

如图 11-1 所示，在“主”标签页上可以设置服务器的基本信息。其中，“服务器名”和“网主电子邮件地址”会在 Apache 显示默认错误页面时使用到。这里输入“au-mel-rhel-1.example.com”和“webmaster@example.com”。“Available Addresses”列表允许指定 Apache 监听哪些请求所在的 IP 地址和端口号，默认为监听所有地址的 80 端口。



图 11-1 Apache 配置界面

下一个标签页包含默认虚拟主机的设置项。我们还没有为任何请求明确地设置自定义的虚拟主机，所以默认虚拟主机将为任何请求提供服务。想要简单了解什么是虚拟主机，请参考补充知识“虚拟主机”，稍后返回“虚拟主机”标签页。

虚拟主机

单台 Linux 主机上运行的 Apache 能为几百个或者几千个网站服务，这些网站都有自己的主机名。我们称它虚拟主机是因为每个站点并不是单个的 Linux 主机。实际中有两种虚拟主机——以 IP 寻址的虚拟主机和以名称寻址的虚拟主机。Apache 对这两者都提供支持。

以 IP 寻址的虚拟主机使 Apache 根据请求的 IP 地址，从特定的目录提供网页服务。对于每个以 IP 寻址的虚拟主机，Linux 主机需要把每个 IP 地址分配给一个网络接口。这是通

过别名实现的，就像在第 6 章中讨论的一样。如果想用安全证书来保护虚拟站点，则需要使用以 IP 寻址的虚拟主机。

以名称寻址的虚拟主机使 Apache 根据 HTTP 请求中的站点名称，从相应的目录中提供网页服务。任意多个以名称寻址的虚拟主机可以共享单个 IP 地址。站点的名称通过发送给 Web 服务器的 HTTP 请求中的一个特定的头来确定。

下一个标签页是“服务器”，如图 11-2 所示。该标签页包含的一些变量用以规定 Web 服务器如何运行。

“锁文件”和“PID 文件”允许一些脚本工具检查 Web 服务器是否正在运行，而“核心转储目录”指定在 Web 服务器严重崩溃的情况下，内存将被转储到哪里。

该标签页中，最重要的是“用户”和“组群”两个域，它们指定 Web 服务器以哪个用户账户运行。记住，在这里不要填写被系统其他地方使用的用户和组，因为这样可能会允许 Apache 访问原本无权查看的文件或目录。

最后，“调整性能”标签页允许用户调节服务器一些基本的限制，如图 11-3 所示。



图 11-2 服务器进程配置



图 11-3 Web 服务器基本的性能优化

“最多连接数量”一项限定服务器可以处理多少个并发连接。许多浏览器同时与 Web 服务器建立多个连接，以便同时下载网页中的样式表和图片，这意味着最大连接数并不等于服务器所能处理的最大用户个数。每个连接也会占用一定数量的系统资源，比如 RAM 和 CPU 时间，所以不要把这个值设置得太高。

“连接超时”设置项规定 Apache 将会等待客户端多长时间，以允许客户端告诉它数据已接收或者向它发送数据。如果 300 秒内客户端没有任何响应，服务器将主动关闭该连接。

由于有些模块可能含有缺陷，不能释放已用资源，因此在此类模块进程为特定数量的连接提供完服务之后，我们就可以让服务器把它销毁，随后它被一个新的服务进程取代。“每次连接最多请求数量”设置项指定了在一个 Web 服务器进程被新进程取代之前，它可以为多少个请求提供服务。

最后，指定服务器是否允许持久性的连接。如果允许，则客户端可以通过同一个连接请求多个文件。通过让服务器和客户端不必为每个被检索的文件建立新的连接，可以提高服务

的速率和效率。

“下次连接的超时时间”设置项指定一个连接闲置多少秒后被服务器关闭。我们在这里暂且保持所有这些设置项的默认值不变，返回“虚拟主机”标签页。

选择“默认虚拟主机”并单击“编辑”按钮，打开“虚拟主机的属性”对话框，如图 11-4 所示。

“虚拟主机名”一项中应设置一个可读的主机名称，以使管理更加方便，它丝毫不会影响 Apache 的工作方式。该对话框中最重要的是“文档根目录”设置项，因为 Apache 服务的文件就来自该目录。我们也可以重写虚拟主机的“网主电子邮件地址”。

“页码选项”标签页（见图 11-5）允许用户指定目录中哪个文件作为默认主页。当客户端请求一个目录时，Apache 检查“Directory Page Search List”搜索列表中的每个文件是否存在，并把找到的第一个文件发送给浏览器。

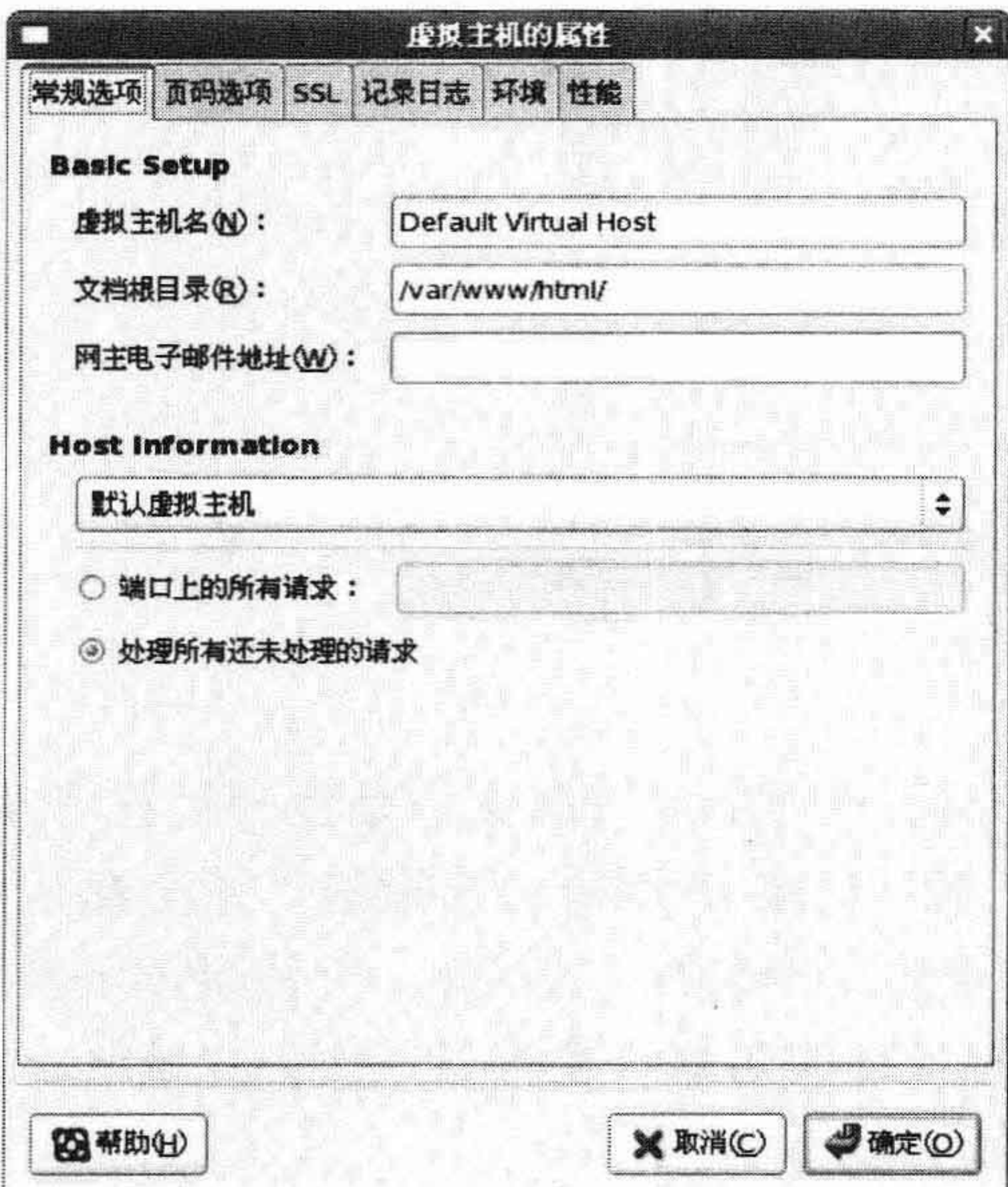


图 11-4 默认虚拟主机设置

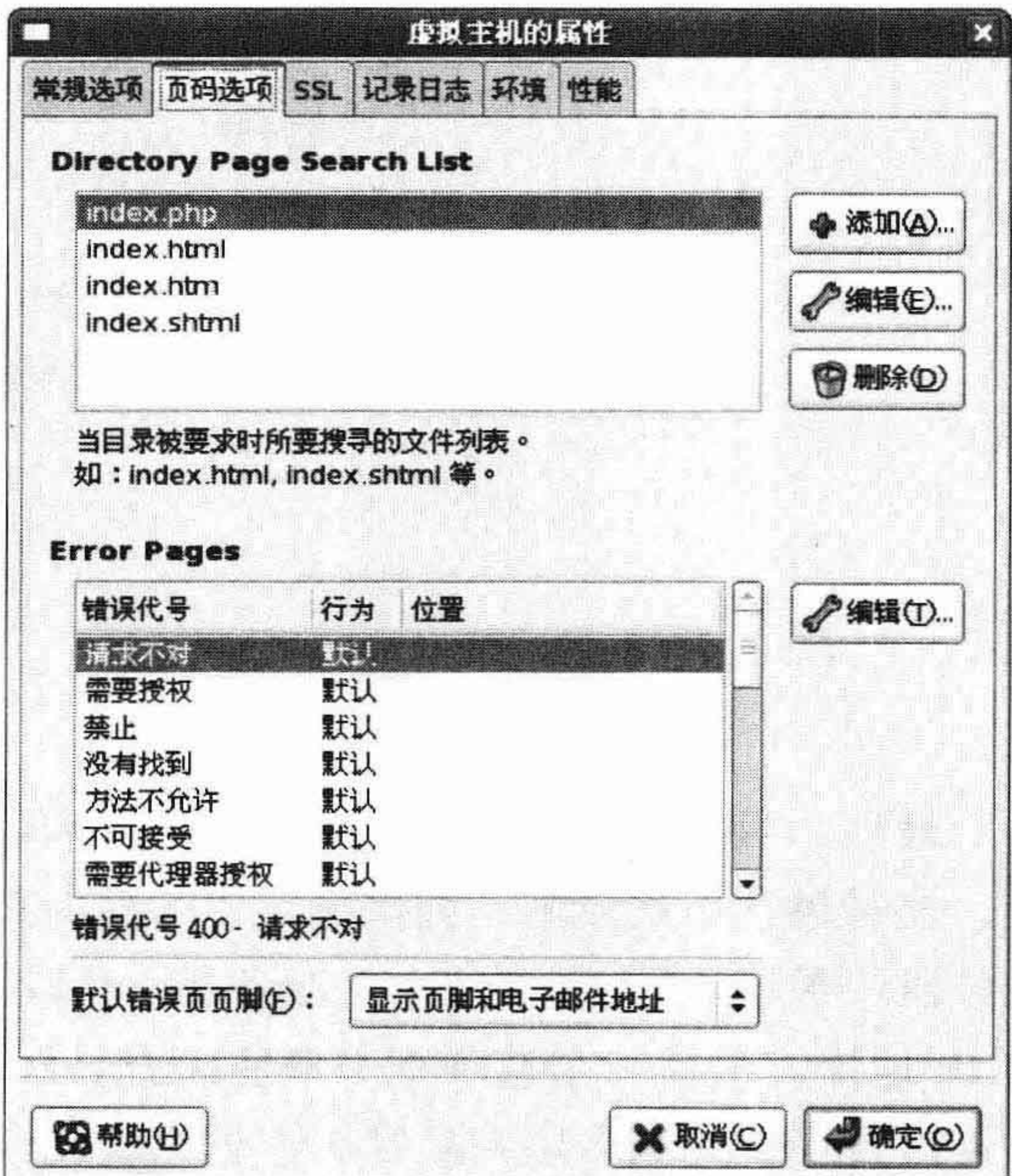


图 11-5 页面选项设置

在本标签页中还可以修改 Apache 的默认错误行为。一般情况下，如果服务器遇到错误，Apache 将显示一个标准的错误页面。如选中“请求不对”项并单击“编辑”按钮，可以使服务器显示一个自定义页面，或者通过一个指定的 URL 把用户重定向到另一个完全不同的站点，如图 11-6 所示。例如，如果用户所请求的文件在服务器上找不到，就可以把用户重定向到一个搜索页面。在此保留其默认行为设置。

“SSL”标签页允许我们配置虚拟主机的安全证书。在稍后配置一个安全的 webmail 站点时，将再次回到该标签页的讨论。

“日志记录”（Logging）标签页（见图 11-7）允许我们更改日志文件。该文件用于记录访问信息和那些可能出现的任何错误信息。如果有多个虚拟主

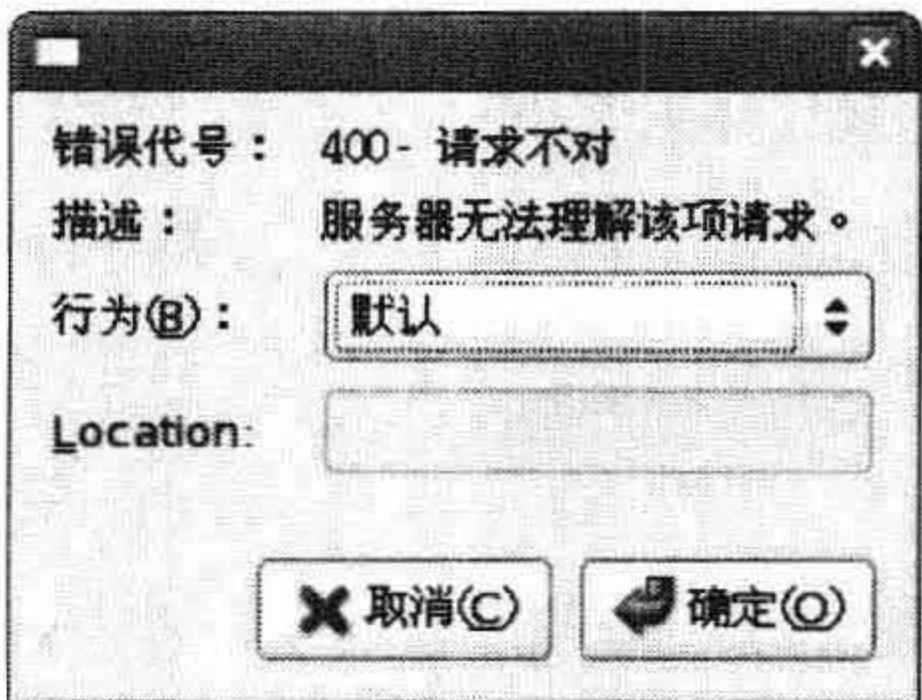


图 11-6 更改出错行为

机，为了便于日志分析工具处理日志信息，可以要求每个虚拟机把各自的日志信息写到不同的文件，还可以通过调整“日志级别”改变日志记录的数量。为了约束错误日志文件的大小，这里设置“日志级别”为 Info，而不是 Debug。

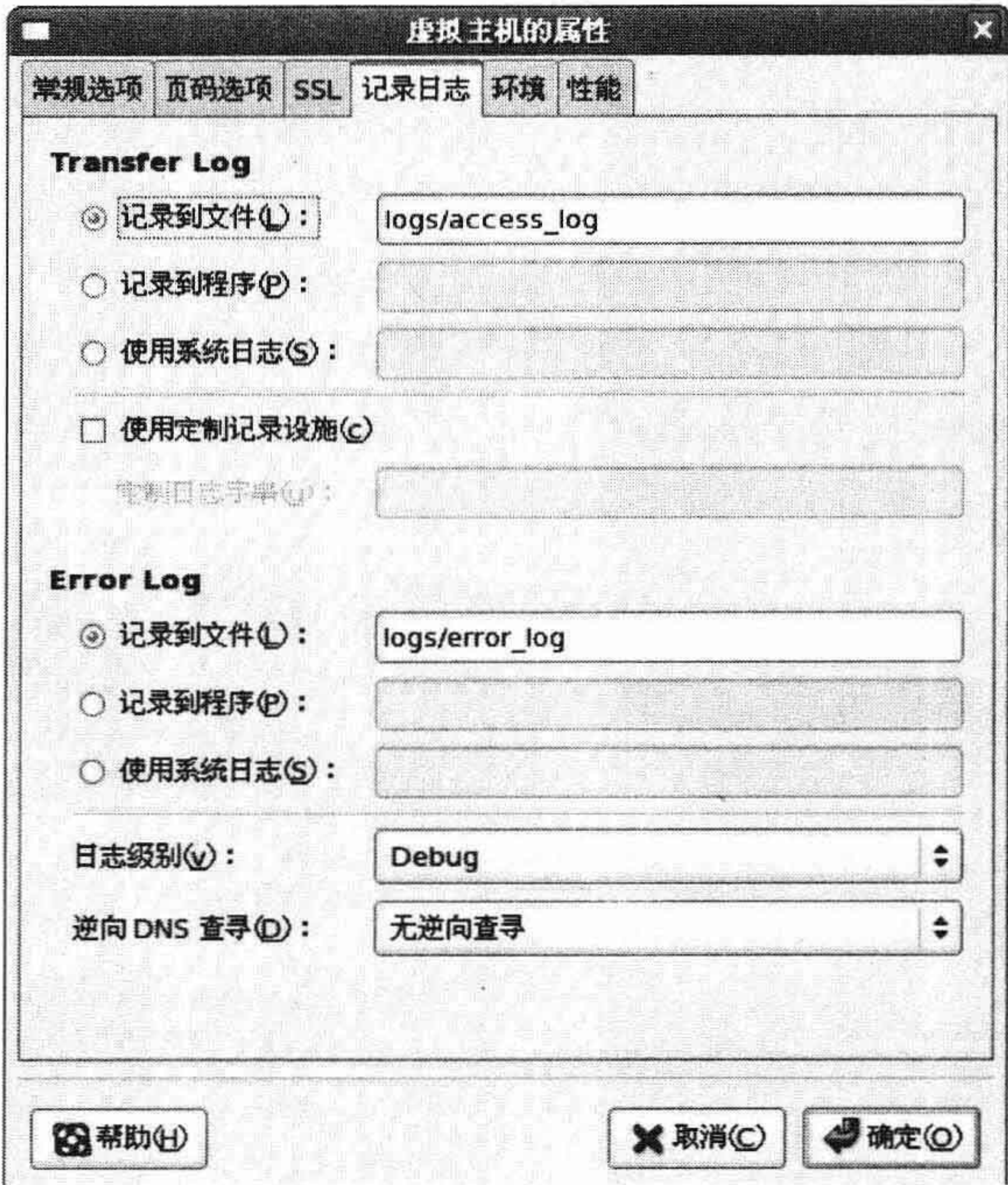


图 11-7 日志配置

最后，“逆向 DNS 查寻” (Reverse DNS Lookup)设置项指定日志文件是包含远程 IP 地址，还是包含正式域名。如果选定“无逆向查寻”，则只记录 IP 地址；当指定“逆向查寻”，服务器将查寻已配置好的 DNS 服务器，获取正在连接的客户端的正式域名，并记录该域名而非 IP 地址。最后一个选项是“双逆向查寻”，该选项促使服务器首先查寻一次正式域名，然后再次查寻正式域名以判断是否能把它转化成原始的 IP 地址。

由于这些查寻执行起来比较慢，开启它们会影响服务器的性能，所以我们设置该选项为“无逆向查寻”。而且，大多数的日志文件分析工具都可以更为高效地进行 DNS 查询。

“环境”标签页允许我们把启动脚本的环境变量传递给 Web 服务器，以便启动脚本可以被 CGI 脚本访问。CGI 表示公共网关接口，它是一个协议。在 Web 服务器接收到浏览器的 URL 请求时，CGI 允许 Web 服务器运行相应的应用程序。这些环境变量可能包含诸如语言设置或时区设置之类的内容。可以在 http://en.wikipedia.org/wiki/Common_Gateway_Interface 上读到关于 CGI 的更多内容。

“性能”标签页允许为虚拟主机和它所包含的特殊目录设置一些 Web 服务器的特定选项。这些选项决定 Web 服务器可以提供哪些文件、是否执行 CGI 脚本，甚至决定 Web 服务器能否通过附加的配置文件重写 Apache 配置选项。在该标签页中，默认生效的选项如表 11-1 所示。

表 11-1 虚拟主机的默认选项

选项	功能
ExecCGI	把脚本当作应用程序来执行，其输出内容被发往浏览器
FollowSymLinks	服务器能解释符号链接，并提供符号链接指向的文件或目录
Includes	服务器执行嵌入 Web 页面的服务器端嵌入指令（Server-side include）
IncludesNOEXEC	服务器端嵌入指令不能在服务器上执行脚本
Indexes	如果目录中没有索引页，则直接用列表显示目录内容
SymLinksIfOwnerMatch	当且仅当符号链接的拥有者与其链接目标的拥有者相同时，服务器才解释符号链接

这里不打算再修改虚拟主机的其他默认配置，单击“确定”按钮关闭“虚拟主机的属性”对话框。

下面建立一个虚拟主机来安置公司网站。我们希望把虚拟主机设置为 `http://www.example.com` 和 `http://example.com`。单击“虚拟主机”标签页，然后单击“添加”按钮。

在新的“虚拟主机的属性”窗口中，设置“虚拟主机名”为 `www.example.com`，“文档根目录”为 `/srv/www/www.example.com/html`，以及“网主电子邮件地址”为 `webmaster@example.com`，如图 11-8 所示。



图 11-8 添加虚拟主机 `www.example.com`

下一步，指定该虚拟主机为一个以名称寻址的虚拟主机。为了让服务器上的所有 IP 地址访问该虚拟主机，在 IP 地址字段中输入“*”。

为了让服务器以名称寻址的方式从该站点提供文件，需要把主机名设置为 `www.example.com`。记住，“虚拟主机名”字段只是一个标签，以方便我们在设置列表中找到它。

最后，因为我们希望通过别名 `example.com` 也能访问该站点，单击“添加”按钮，把 `example.com` 添加到别名列表，然后单击“确定”按钮。

保持“页码选项”标签页的设置项不变，并且不使用该虚拟主机的 SSL 服务。在“记录日志”标签页，设置“日志级别”为 `Error`，确保服务器不执行任何逆向 DNS 查寻。同样，不需要修改“环境”选项，但我们要防止服务器执行虚拟主机根目录中的 CGI 脚本。为此，单击“性能”标签页，然后单击“添加”按钮，为新的根目录添加一些设置。在该列表中，不要选中“ExecCGI”复选框，勾选“让 `.htaccess` 文件取代目录选项”这一复选框，如图 11-9 所示。

单击“确定”按钮接受更改，接着再次单击“确定”按钮关闭“虚拟主机的属性”对话框。并再次单击“确定”按钮关闭“HTTP”配置工具。当弹出询问对话框，单击“是”接受所做的更改。

现在剩下的事就是要确保根目录是存在的，否则当尝试访问站点 `http://www.example.com` 时将产生错误。给 `mkdir` 命令传递 `-p` 选项，就可以通过该命令来创建根目录。`-p` 选项确保 `mkdir` 命令创建任何所需的父目录。

```
$ sudo mkdir -p /srv/www/www.example.com/html
```

在 Red Hat 上，Web 服务器在安装后并没有启动，`init.d` 目录中的符号链接也没有设置。需要用 `chkconfig` 命令手动完成这些操作。

```
$ sudo chkconfig --level 2345 httpd on
```

在启动 Web 服务器之前，还要重新配置防火墙，使之允许 Web 服务器的端口传输信息。选择“系统”>“管理”>“安全级别和防火墙”，勾选“WWW(HTTP)”复选框，如图 11-10 所示。接着单击“确定”按钮关闭该程序，然后单击“是”按钮让更改生效。

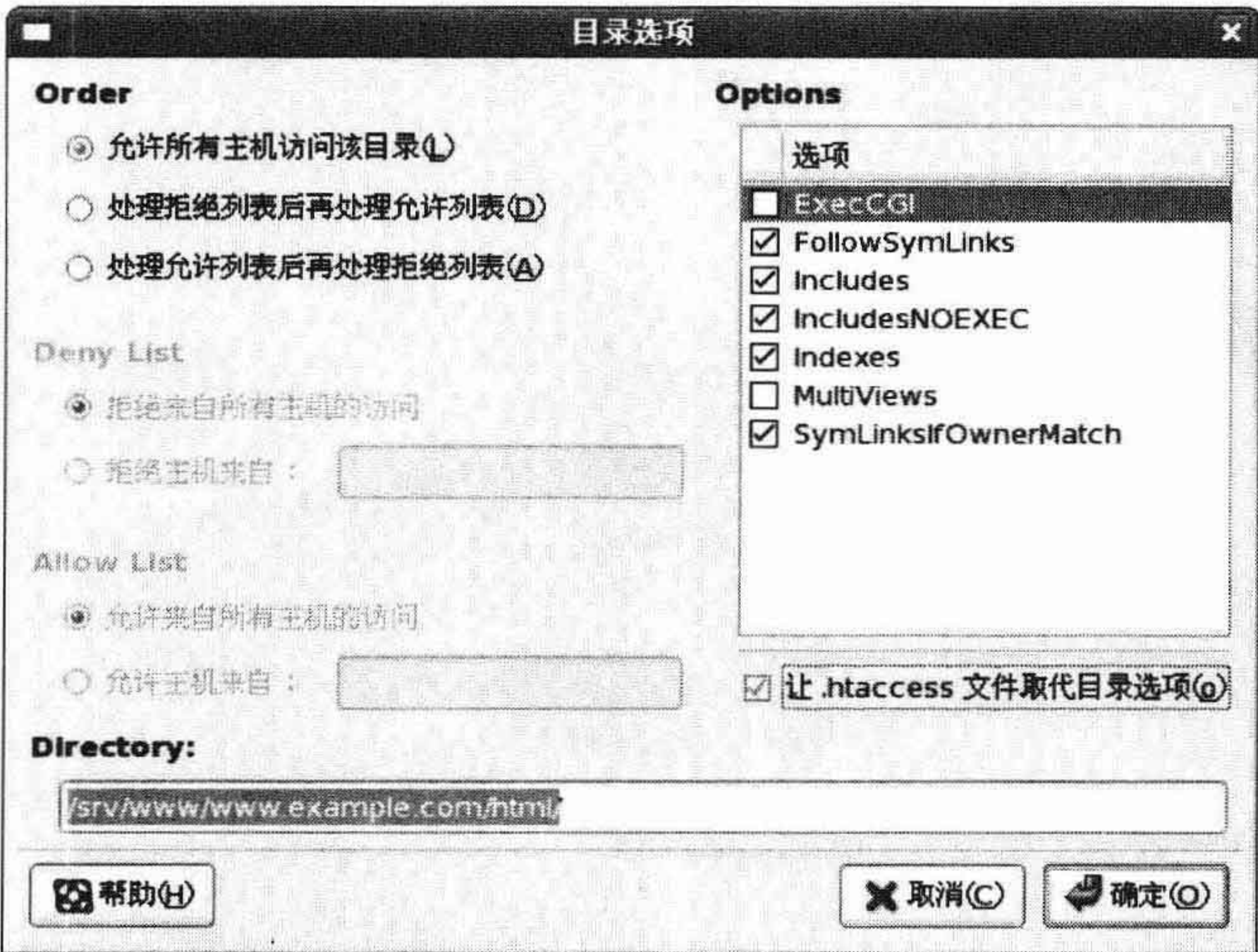


图 11-9 更改目录选项



图 11-10 允许 Web 服务器穿透防火墙

现在我们可以用 `service` 命令启动 Apache 服务器。

```
$ sudo service httpd start
```


如果在浏览器中访问 `http://localhost/`，将会看到一个普通的欢迎页面。

Apache 的模块配置和稍高级的配置可以手动完成。可以直接修改主配置文件，或者在 `/etc/httpd/conf.d` 目录中添加配置文件片段。主配置文件引用了 `/etc/httpd/conf.d` 目录。稍后将返回“Modules”部分的模块配置。

无论何时修改了配置都必须让 Apache 重新加载配置文件。可以用“`sudo apachectl graceful`”重新加载配置文件，而不是每次都先停止服务器，然后再启动服务器，该命令告诉 Web 服务器无需切断所有的活动连接就可以重新加载配置文件。通过运行“`sudo apachectl configtest`”，还可以用 `apachectl` 命令检查配置文件中的错误。如果习惯了这么做，就永远不会因为错误的配置指令而造成服务器崩溃。

2. Ubuntu

在 Ubuntu 上，不能通过图形用户界面的方式管理 Apache 的配置项，而且基本的配置项被拆分到多个配置文件中。在 Ubuntu 上，Apache 加载的主配置文件是 `/etc/apache2/apache2.conf`。该文件包括了列表 11-1 中的基本配置指令。除了引用附加文件来配置虚拟主机、模块、端口和 IP 地址，以及引用 `/etc/apache2/conf.d` 目录中的配置片段，主配置文件仅含有设置日志和性能的指令。

列表 11-1 Ubuntu 上的 Apache 默认配置 `apache2.conf`

```
ServerRoot "/etc/apache2"
LockFile /var/lock/apache2/accept.lock
PidFile ${APACHE_PID_FILE}

Timeout 300
KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 15

<IfModule mpm_prefork_module>
    StartServers      5
    MinSpareServers   5
    MaxSpareServers   10
    MaxClients        150
    MaxRequestsPerChild 0
</IfModule>

User ${APACHE_RUN_USER}
Group ${APACHE_RUN_GROUP}

AccessFileName .htaccess
<Files ~ "^\.ht">
    Order allow,deny
    Deny from all
</Files>

HostnameLookups Off
ErrorLog /var/log/apache2/error.log
LogLevel warn
Include /etc/apache2/mods-enabled/*.load
```



```
Include /etc/apache2/mods-enabled/*.conf
Include /etc/apache2/httpd.conf
Include /etc/apache2/ports.conf
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent
ServerTokens Full

ServerSignature On
Include /etc/apache2/conf.d/
Include /etc/apache2/sites-enabled/
```

Ubuntu 上的默认配置和 Red Hat 上的大部分相同，只是默认的文件位置不一样，而且不能以图形用户界面的方式来配置和管理虚拟主机。

要启动的服务器实例个数被定义在 `IfModule` 标签所包围的特殊区块中。当特定的模块被载入服务器时，可以通过相应的标签设置一些配置变量。本例中，如果模块 `mpm_prefork_module` 被载入，服务器的实例个数将被设置。而当 `mpm_prefork_module` 模块不存在时，Apache 就不会识别这些变量并且错误退出，所以该模块生效是有条件的。

`AccessFileName` 指令也很重要，它指定 Apache 指令配置文件的名称。在列表 11-1 中，该文件被命名为 `.htaccess`，任何 Web 目录都可以包含这样的文件。在 Apache 试图为正在连接的客户端提供服务之前，它将检查 `.htaccess` 文件是否存在，如果存在则处理它所包含的指令。

紧接着 `AccessFileName` 下面的指令能防止远程用户下载任何文件名以 `.ht` 开头的文件。需要该指令是因为 `.htaccess` 文件可能含有被骇客利用的潜在信息。

`Order` 指令决定控制文件访问的方式。参数“`allow, deny`”首先列出允许访问指定文件的所有主机，然后列出不允许访问的主机。在本例中，没有明确地允许任何人访问文件，而是用“`Deny from all`”指令明确地拒绝所有人的访问。

前 4 个 `Include` 指令让 Apache 处理所指定文件中的配置信息。为了加载激活的模块，主配置文件需包含 `mods-enabled/*.load`。这些模块的配置信息随后从 `mods-enabled/*.conf` 文件中读取。因为一些自动安装脚本会往 `httpd.conf` 文件中写入信息，所以 `httpd.conf` 也被包含进主配置文件（由于历史遗留的原因，仍保留 `httpd.conf` 文件）。最后，Apache 从文件 `ports.conf` 载入所要监听的 IP 地址和端口的配置信息。

`ServerTokens` 指明 Apache 能向客户端公开多少信息，这些信息包括 Apache 的版本、运行所在的操作系统及启用的模块等。`ServerTokens` 设置为 `Full` 意味着 Apache 通过 `Server header` 字段公开所有信息。我们可以用 `HEAD` 工具查看 `Server header` 字段。在 Red Hat 上，该工具的软件包是 `perl-libwww-perl`；在 Ubuntu 上，相应的软件包是 `libwww-perl`。`HEAD` 工具接受一个 URL 作为参数，然后显示它接收到的 HTTP 的头信息。

```
$ HEAD http://localhost | grep Server:
Server: Apache/2.2.8 (Ubuntu)
```

为了更改 HTTP 头信息，可以用 `Pro` 替换 `Full`，并使用“`sudo apache2ctl reload`”命令让 Apache 重新加载配置文件。当再次检查 `Server header` 字段信息时，版本号和操作系统信息将不再出现。


```
$ HEAD http://localhost | grep Server:
Server: Apache
```

ServerSignature On 指令将使 Apache 把一些基本的服务器信息添加到默认错误页面。当服务器遇到错误，就会显示该错误页面。

最后，Apache 会从 `conf.d` 目录下的文件中载入所有的自定义配置片段，从 `sites-enabled` 目录的文件中读取虚拟主机的全部信息。

在 Ubuntu 上，默认虚拟主机被定义在 `/etc/apache2/sites-available/default` 中，并且设定 `/var/www` 作为其基础根目录。如果在浏览器中访问 Ubuntu 主机的地址 “`http://localhost/`”，将会看到简洁的 “It works!” 页面。

为了在 Ubuntu 上添加虚拟主机，我们在 `/etc/apache2/sites-available` 目录中创建一个虚拟主机定义文件。然后运行 “`sudo a2ensite`” 命令，使该文件对 Apache 有效。“`sudo a2ensite`” 会在 `/etc/apache2/sites-enabled` 目录中为该定义文件创建一个符号链接。

下面为 `www.example.com` 站点添加一个虚拟主机，并将它存储在之前创建的逻辑卷中。在添加虚拟主机之前，在磁盘中为虚拟主机选择一套系统的存储方案是很有用的。随着主机中站点数目的增多，这将使管理变得简单。

为了设置该虚拟主机，需要把列表 11-2 中的指令添加到 `/etc/apache2/sites-available/www.example.com` 文件中。

列表 11-2 新的虚拟主机设置

```
<VirtualHost *>
    ServerName www.example.com
    ServerAlias example.com
    ServerAdmin webmaster@localhost
    DocumentRoot /srv/www/www.example.com/html
    <Directory /srv/www/www.example.com/html>
        Options Indexes FollowSymLinks Includes IncludesNOEXEC SymLinksIfOwnerMatch
        AllowOverride All
    </Directory>
</VirtualHost>
```

正如我们所看到的，在 Apache 的配置中使用了类似于 HTML 的标签。首先，打开 “VirtualHost” 标签，并指定该定义适用于所有地址。接着，指定该虚拟主机的名称。任何给定的虚拟主机只能拥有一个 `ServerName`，但是可以通过 `ServerAlias` 指令添加其他名称也可以用更多的 `ServerAlias` 指令为虚拟主机添加额外的别名。

`DocumentRoot` 指令指定该虚拟主机将从哪个目录提供文件。当该目录被设定时，我们可以用 “Directory” 标签为该目录以及它所包含的所有文件和目录指定配置项。通过设置 `AllowOverride` 为 `All`，允许服务器用 `.htaccess` 文件中的配置更改 Apache 的设置项。设置 `AllowOverride` 为 `None`，则禁止使用 `.htaccess` 文件。最后，关闭 “Directory” 和 “VirtualHost” 标签，并保存该文件。

与 Red Hat 相反，Ubuntu 必须明确地启用虚拟主机，这就需要在 `/etc/apache2/sites-enabled` 中创建一个链接，以链接到刚刚创建的配置片段。幸运的是，不需要用 `ln` 命令手动创建链接。`apache2-common` 软件包提供了一个方便的工具来完成该操作。该工具名叫 `a2ensite`，它与

apache2 一起被自动安装。而且我们可以把配置片段的文件名作为参数来运行 a2ensite。

```
$ sudo a2ensite www.example.com
Site www.example.com installed; run /etc/init.d/apache2 reload to enable.
```

现在，可以按照以上脚本所建议的进行操作，或者使用“`sudo apache2ctl graceful`”命令手动地重新加载 Apache 服务器配置。在 Apache 服务器重启之前，Apache 的启动脚本首先运行“`apache2ctl configtest`”命令检查配置文件是否正确。启动服务器之前检查配置文件正确与否是个好习惯，即便在使用 `apache2ctl` 命令重新加载服务器配置之前也最好用“`apache2ctl configtest`”命令检查一下配置文件是否存在错误。

■注：Ubuntu 也有 `apachectl` 命令。该命令是 Apache 历史遗留版本 1.3 版的软件包的一部分。

11.1.2 访问控制

之前曾提到可使用 `.htaccess` 文件中带有 `Allow` 或者 `Deny` 参数的 `Order` 指令来进行访问控制。例如，要限制只有局域网内的主机可以访问站点 `http://www.example.com`，可在 `/srv/www/www.example.com/html` 目录中创建一个 `.htaccess` 文件，并且添加以下指令。

```
Order allow,deny
Allow from 192.168.0.0/255.255.255.0
Deny from all
```

上述指令告诉 Apache 首先检查那些可能用指令所设置的网络地址和子网掩码访问该站点的主机。接着使用 `all` 关键字指示 Apache 拒绝所有来自其他主机的访问。通过反转两个参数的顺序，还可以只拒绝局域网内其他主机的访问。

```
Order deny,allow
Deny from 192.168.1.0/255.255.255.0
Allow from all
```

以上指令首先拒绝了来自所设置地址的主机的访问，然后接受来自其他地址的主机访问。

另一种控制资源访问的方法是要求用户输入用户名和密码。许多 Web 应用程序本身就实现了这种方法，但是也可以让 Apache 管理一个用户名和密码列表，这样我们不需要额外的软件就能保护特殊的目录。

首先，需要用 `htpasswd` 工具创建一个包含需要用到的用户名和密码的文件。通常，把要用到的文件名和需要创建的用户名作为参数传递给 `htpasswd`，如果该文件尚未存在，则还需要加上 `-c` 选项。

```
$ sudo htpasswd -c /srv/www/www.example.com/htpasswd jsmith
New password:
Re-type new password:
Adding password for user jsmith
```

现在，不需要用到 `-c` 选项也可以添加其他用户。如果不想被提示输入密码，我们还可以通过 `-b` 选项在命令行中设置它。


```
$ sudo htpasswd -b /srv/www/www.example.com/htpasswd ataylor s3kr@t
Adding password for user ataylor
```

接下来，需要告诉 Apache 在接受请求时要求身份认证。只要重写 AuthConfig 选项，就可以在.htaccess 文件中实现身份认证，如列表 11-3 所示。因为设置了 AllowOverride 为 All，所以列表 11-3 中的指令将不会出现问题。

在列表 11-3 中，首先用 AuthType 指令指定想要使用的认证的类型。在本例中，AuthType 被设置为 basic。然后，需要用 AuthBasicProvider 指令告诉 Apache 用哪个模块提供基本的认证功能。本例中，它被设置为 file。接着，通过 AuthUserFile 指令告诉 Apache 哪个文件保存有认证信息。

为了帮助用户明确他们试图访问的内容，可以通过 AuthName 指令为受保护的资源指定一个名称。当用户被要求输入认证信息时，该名称将显示给他们，如图 11-11 所示。所以，最好准确地描述该名称。

最后，需要告知 Apache 只有当用户认证成功时才授予其访问权限。可以通过指令 “Require valid-user” 做到这一点。

列表 11-3 .htaccess 文件中的认证配置

```
AuthType basic
AuthBasicProvider file
AuthUserFile /srv/www/www.example.com/htpasswd
AuthName "Restricted Area"
Require valid-user
```

如果现在访问 http://www.example.com，浏览器将要求输入用户名和密码，如图 11-11 所示。

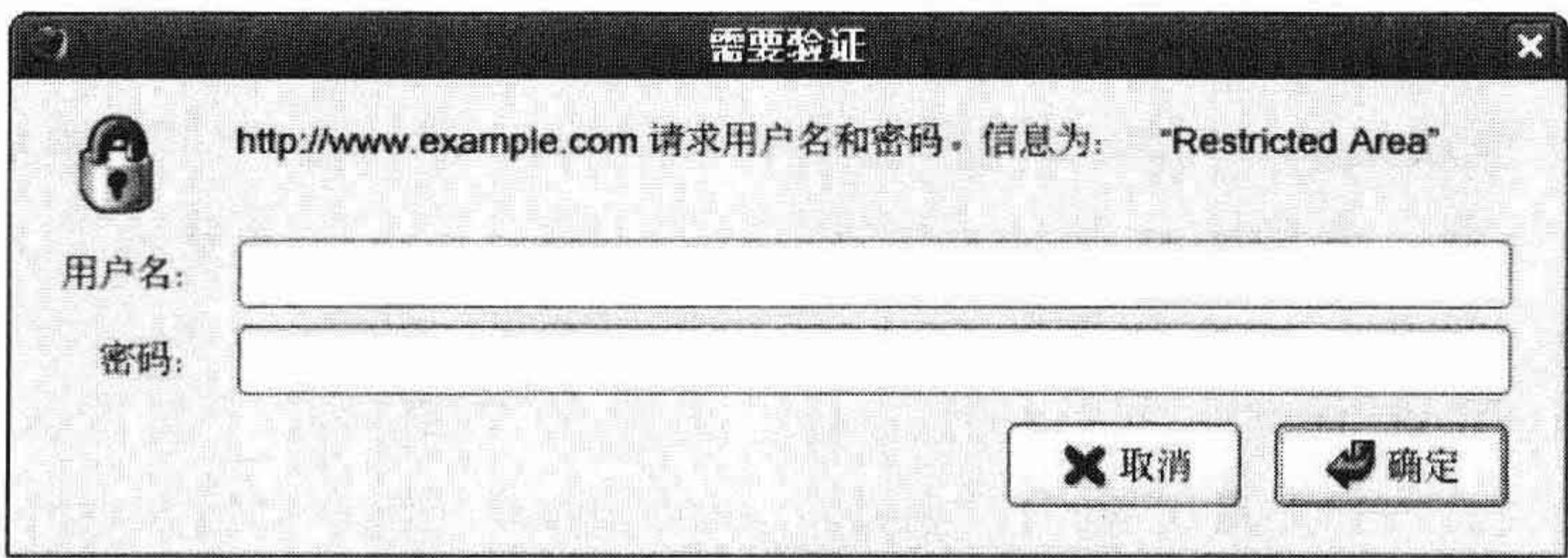


图 11-11 Apache 用户认证

如果无法提供有效的用户名和密码，则不被授予该站点的访问权；如果输入有效的认证信息，Apache 就会允许浏览该站点。

在 Apache 的文档网站上，可以读到更多关于基于虚拟主机的访问控制和基于用户的访问控制的内容：

- <http://httpd.apache.org/docs/2.2/howto/access.html>
- http://httpd.apache.org/docs/2.2/mod/mod_auth_basic.html

11.1.3 模块

模块能给 Apache 提供附加功能。模块由 LoadModule 指令激活，该指令指定了将要加载

的模块文件的路径。

在 Red Hat 上，附加模块通常由 `/etc/httpd/conf.d` 目录中的配置片段激活。这些配置片段随同模块软件包一起安装。当重启 Apache 时，它识别这些新的文件并处理文件中的指令。要想不包含某个配置片段，也就是禁用相应的模块，可以重命名该片段，使它的文件名不再以 `.conf` 结尾。

在 Ubuntu 上，模块软件包在 `/etc/apache2/mods-available` 目录中添加这些配置片段，然后在 `/etc/apache2/modules-enabled` 目录中创建它们的链接。这些链接也可以使用 `a2enmod` 和 `a2dismod` 命令手动地创建，方法类似于 `a2ensite` 和 `a2dissite`。

例如，需要激活 `rewrite` 模块（该模块允许 Apache 重写 URL），让它可以被随后安装的一个网站使用，则可以通过运行 `sudo a2enmod rewrite` 命令，重新加载服务器的配置，这样 `rewrite` 模块就被激活了。

可以在 <http://httpd.apache.org/docs/2.2/mod/> 上获得关于 Apache 包含的所有模块的信息，以及各模块所提供的功能信息。

安装 PHP 模块

许多 Web 应用程序是用 PHP 写的。PHP 是 Rasmus Lerdorf 开发的一门脚本语言。当浏览器从这样的站点请求一个页面时，Web 服务器处理页面中的代码并把输出显示到浏览器。为了能安置这些 Web 应用程序，Web 服务器要能够解析并且执行 PHP 代码。

对于某些 Web 服务器，上述过程是通过 CGI 脚本来处理的，但是在 Apache 上，我们可以通过安装一个模块直接给服务器添加 PHP 模块。

■注：可以在 <http://www.php.net/> 读到关于 PHP 的所有内容。

PHP 自身就是模块化的，可以安装附加的软件包为 PHP 添加功能。稍后我们将建立一个 MySQL 数据库服务器，而为了让 Web 应用程序可以使用它，还需要为 PHP 添加 MySQL 支持。另外，我们将为使用广泛的图形库 GD、字符串转换库 `mbstring` 以及电子邮件协议 IMAP 安装支持软件包。IMAP 还允许安装和使用基于 PHP 的 Web 版电子邮件应用程序。

在 Red Hat 上，可以通过命令 “`sudo yum install php5 php-mysql php-gd php-imap php-mbstring`” 安装所有这些库；而在 Ubuntu 上，则通过命令 “`sudo aptitude install libapache2-mod-php5 php5-mysql php5-gd php5-imap`” 进行安装。在 Red Hat 上 PHP 默认是激活的，但是在 Ubuntu 上，需要运行 “`sudo a2enmod php5`” 命令来激活它。不过在这两个系统上，为了使 PHP 生效，都需要重启 Apache。

11.1.4 文件和目录许可

处理站点时，需要站点所在的根目录的写权限。但是，Apache 用户不能往这些目录中写东西，因为如果匿名的 Web 用户在站点中找到漏洞，这种权限可能会允许它们往系统中写文件。

如果多个用户同时管理站点，那么最好创建一个用户组。只要该用户组拥有根目录的写权限，添加到该组中的任何用户都可以往根目录中写文件和创建目录。

■注：使用一个特定的系统用户组来管理网站，还意味着我们可以允许该组中的用户使用 `apachectl` 或者 `apache2ctl` 命令——通过和 `sudo` 结合——而不需要给他们 `root` 用户的全部权限。

为了确保一个用户所创建的文件能够被同一组中的另一个用户更改，需要设置 `umask`，以便新创建的文件和目录能被同组中的用户改写。还需要设置 `setgid` 位，让新的文件和目录从父目录的用户组中继承所有权，而不是从创建该文件或者目录的用户的原始用户组继承所有权。稍后，在安装一些 Web 应用程序时，我们将就此举个例子。

■注：可以在 <http://httpd.apache.org/> 上获取更多的信息和 Apache 相关文档。

11.2 MySQL 数据库

因为许多基于 Web 的应用程序都使用 MySQL 数据库存储数据，所以我们还将展示如何安装 MySQL 数据库服务器。不同于 Microsoft Access，MySQL 由服务器和客户端组成。MySQL 服务器提供数据存储和检索服务，而客户端可以是任何使用 MySQL 服务器的应用程序——命令行工具、OpenOffice 或者一个被网站使用的库。

11.2.1 安装

在 Red Hat 和 Ubuntu 两个系统上，MySQL 服务器组件是由 `mysql-server` 软件包提供的。可以分别通过运行“`sudo yum install mysql-server`”和“`sudo aptitude install mysql-server`”安装它。在 Ubuntu 上，安装进程将要求输入 MySQL 的 `root` 用户密码以确保 MySQL `root` 账号的安全性。而在 Red Hat 上，我们需要手动执行这一步骤。

1. Red Hat

在 Red Hat 上，需要更改一些基本配置。默认情况下，MySQL 服务器在所有已配置的网络接口和地址上监听连接。这样并不安全，所以我们将限制它只在回环地址上监听。

在文本编辑器里打开 `/etc/my.cnf`，并且在 `[mysqld]` 区块下添加以下一行。

```
bind-address = 127.0.0.1
```

不要使用旧密码格式，所以注释掉指示旧密码格式的这行。

```
# old_passwords=1
```

新的配置文件如列表 11-4 所示。

列表 11-4 Red Hat 上新的 `/etc/my.conf` 文件

```
[mysqld]
datadir=/var/lib/mysql
```



```
socket=/var/lib/mysql/mysql.sock
user=mysql

# Default to using old password format for compatibility with mysql 3.x
# clients (those using the mysqlclient10 compatibility package).
# old_passwords=1
bind-address=127.0.0.1

[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
```

现在，可以通过 `service` 命令启动 MySQL 服务器。首次启动 MySQL 服务器时，它将配置一些默认的表，并且提醒我们设置一个 root 密码，如列表 11-5 所示。

列表 11-5 Red Hat 上 MySQL 首次运行

```
$ sudo service mysqld start
Initializing MySQL database: Installing MySQL system tables...
OK
Filling help tables...
OK
...
```

现在 MySQL 服务器已经开始运行了，可以设置一个 root 密码并清除那些默认的表。MySQL 服务器带有一个称作 `mysql_secure_installation` 的实用工具，它将为我们做这些工作，所以无需按照启动脚本的提示手动地运行 `mysqladmin` 命令。

```
$ sudo mysql_secure_installation
```

```
NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MySQL
SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!
```

```
In order to log into MySQL to secure it, we'll need the current
password for the root user. If you've just installed MySQL, and
you haven't set the root password yet, the password will be blank,
so you should just press enter here.
```

```
Enter current password for root (enter for none):
OK, successfully used password, moving on...
```

```
Setting the root password ensures that nobody can log into the MySQL
root user without the proper authorisation.
```

```
Set root password? [Y/n] y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
... Success!
```

```
By default, a MySQL installation has an anonymous user, allowing anyone
to log into MySQL without having to have a user account created for
them. This is intended only for testing, and to make the installation
go a bit smoother. You should remove them before moving into a
```


production environment.

```
Remove anonymous users? [Y/n] y
... Success!
```

Normally, root should only be allowed to connect from 'localhost'. This ensures that someone cannot guess at the root password from the network.

```
Disallow root login remotely? [Y/n] y
... Success!
```

By default, MySQL comes with a database named 'test' that anyone can access. This is also intended only for testing, and should be removed before moving into a production environment.

```
Remove test database and access to it? [Y/n] y
- Dropping test database...
... Success!
- Removing privileges on test database...
... Success!
```

Reloading the privilege tables will ensure that all changes made so far will take effect immediately.

```
Reload privilege tables now? [Y/n] y
... Success!
```

Cleaning up...

All done! If you've completed all of the above steps, your MySQL installation should now be secure.

Thanks for using MySQL!

现在服务器安全了，它不会接受来自远程主机的连接，而且也不允许没有 MySQL 账号的用户进行连接。可以用 `chkconfig` 命令把 MySQL 服务器的启动脚本添加到相应的目录，使其开机自动运行。

```
$ sudo chkconfig --level 2345 mysqld on
```

2. Ubuntu

在 Ubuntu 中，启动目录中相应的链接已被自动创建，而且 MySQL 服务器在安装后就是运行的。

■注：在 Ubuntu 上，也可以运行 `mysql_secure_installation` 命令整理 `privilege` 表和删除 `test` 表。

11.2.2 测试服务器

为了检查 MySQL 是否正在运行，可以用命令行客户端来连接它。需要用 `-u` 选项指定以

哪一个用户账户来连接 MySQL 服务器。使用 **-p** 选项将提示我们输入相应的密码。

```
$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 5.0.51a-3ubuntu5.4 (Ubuntu)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> SELECT VERSION();
+-----+
| VERSION() |
+-----+
| 5.0.51a-3ubuntu5.4 |
+-----+
1 row in set (0.00 sec)

mysql> \q
Bye
```

能够连接服务器并且运行查询语句表明 MySQL 服务器运行良好。注意，在读者的主机上，返回的版本号字符串可能不一样，因为这取决于所安装的 MySQL 服务器版本。

11.2.3 基本的 InnoDB 优化技术

当创建数据库时，MySQL 允许指定数据库要使用的存储引擎。通过选择存储引擎，就能选择它在磁盘上存储数据的方式。如果不修改配置，MySQL 将使用 MyISAM 存储引擎，除非在创建数据库时明确地选择另外一个。

MyISAM 引擎已经存在很长时间了，但是它却不支持事务和外键这些功能。它从故障（比如系统崩溃）中自动恢复的能力也不是很好，所以我们将设置一个不同的引擎作为 MySQL 服务器的默认存储引擎。

如果没有指定 InnoDB 存储引擎，要配置 MySQL 服务器使用 InnoDB 存储引擎。InnoDB 支持事务和外键，并且即使经历了系统崩溃，它产生问题的可能性也不大。

■注：在 http://tag1consulting.com/MySQL_Engines_MyISAM_vs_InnoDB 上，可以阅读一些资料，在 MyISAM 和 InnoDB 之间做出选择。

在更换默认存储引擎的同时，还将对服务器配置做一些更改以提高其性能。通过调整一些变量可以控制 MySQL 服务器能使用多少的 RAM、如何把已更改的数据写入磁盘以及多久写一次数据。

InnoDB 如何存储数据

InnoDB 有个双层的存储方案。当数据表中的数据发生了更改（比如执行一条更新语句），InnoDB 便在事务日志里面存储这些更改。当事务日志满后，MySQL 服务器就会把这些更改了的数据记录刷新到数据表文件。

这样做是为了提高性能，就和日志文件系统一样。通过一次性执行这些操作，其他时候

磁盘就不需要再为这些操作花费太多时间来寻道和查找数据了。

MySQL 服务器关闭时并不会处理这些事务日志，所以这些文件中通常还含有没有刷新到数据库中的数据。这意味着不能用删除文件和新建文件的方法来增大或减少这些事务日志文件的大小。

我们需要更改事物日志的文件大小。对现在的系统来说，5MB 的默认值实在太小了。事物日志文件很快就会被写满，这意味着 MySQL 服务器需要频繁地清空它们，从而降低服务器的性能。

在更改 InnoDB 的事务日志文件大小之前，需要确保这些事务日志文件不再含有重要数据。停止服务器时，可以强制服务器处理事务日志中的所有记录，并把它们回写到数据表文件中，从而保证日志文件不再含有重要数据。该行为是由变量 `innodb_fast_shutdown` 控制的。服务器正在运行时，以 root 用户连接服务器并且执行查询语句“`SET GLOBAL innodb_fast_shutdown=0`”，就可以更改 `innodb_fast_shutdown` 变量的值，如列表 11-6 所示。

列表 11-6 停止服务器时强制刷新 InnoDB 事务日志

```
$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 32
Server version: 5.0.45 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> SET GLOBAL innodb_fast_shutdown=0;
Query OK, 0 rows affected (0.00 sec)

mysql> \q
Bye
```

现在，可以停止 MySQL 服务器了。MySQL 服务器将把所有挂起的数据更改从事务日志写入到数据表文件。这意味着可以安全地把这些已有的日志文件移出，并且更改事务日志文件大小。在 Red Hat 上，通过“`sudo service mysqld stop`”命令来终止 MySQL 服务器的运行；在 Ubuntu 上，则可使用命令“`sudo invoke-rc.d mysql stop`”。

这些日志文件名为 `ib_logfile0` 和 `ib_logfile1`，可以在 `/var/lib/mysql` 目录中找到它们。我们将把这 2 个文件移出该目录，因此下次启动服务器时，MySQL 会创建新的日志文件。

```
$ cd /var/lib/mysql
/var/lib/mysql$ sudo mv ib_logfile* /root
```

注：不要删除这些日志文件，直到已经证实 MySQL 服务器使用了新的配置。

现在，可以编辑配置文件了。在 Red Hat 上，该文件是 `/etc/my.cnf`，而在 Ubuntu 上则是 `/etc/mysql/my.cnf`。在 Red Hat 上，该文件内容并不完整，因此读者可以把我们给出的配置指令添加到该文件；而在 Ubuntu 上，该文件已经包含了大量的注释和配置指令。读者需要在适当的地方用我们提供的配置替换该配置文件中相应的配置。

在此所做的所有改变都位于 `[mysqld]` 区块下面。首先，更换默认存储引擎。


```
default-storage-engine = InnoDB
```

接下来，更改 InnoDB 存储引擎的行为方式。

```
innodb_log_group_home_dir = /var/lib/mysql
innodb_data_home_dir = /var/lib/mysql
innodb_data_file_path = ibdata1:10M:autoextend
innodb_log_file_size = 32M
innodb_log_buffer_size = 4M
innodb_log_files_in_group = 2
innodb_buffer_pool_size = 128M
innodb_flush_method = O_DIRECT
innodb_file_per_table = 1
```

在此，明确定义了在哪里存放 InnoDB 数据和事务文件，接着定义了 InnoDB 默认数据文件的文件名和大小。首次启动 MySQL 时，10MB 的 ibdata 文件将被创建，所以不需要更改这里所示范的这些值。

我们设置 InnoDB 事务日志的文件大小为 32MB，内存中的日志缓冲区大小为 4MB。这些值意味着服务器会多使用一些 RAM，同时少一些磁盘访问，从而会有更好的性能。同时 innodb_log_files_in_group 变量告知服务器它可拥有两个事物日志文件。

下一步，需要给 MySQL 服务器分配一些 RAM，让它用来保存数据表信息和执行查询。这个数量是由变量 innodb_buffer_pool_size 控制的，我们把它设置为 128MB。在现代运行 MySQL 和其他服务的服务器上，128MB 的大小应该是个合理的值。如果服务器没有 1GB 的 RAM，可能要把该值设置低一点。

通过设置 innodb_flush_method 变量，可以告诉 MySQL 服务器不要在系统磁盘的缓存中缓存任何数据，而要在预留给 InnoDB 缓冲池的内存中存储数据。接着指定 O_DIRECT 变量，防止系统在 RAM 上对数据保留两份副本。最后，设置 innodb_file_per_table 变量让服务器用不同的文件存储数据库的每个表。如果没有指定这个，所有数据会被添加到 ibdata 文件，结果将导致 ibdata 文件无限制地增大。

当数据不在 RAM 中，而需要从磁盘上读取时，MySQL 默认一次读取 128KB 的数据。这节约了内存使用，但在超过 1MB 的数据需要被读取时却很慢。我们设置 read_buffer_size 和 read_rnd_buffer_size 两个变量的大小，以增大该数据块的大小。

```
read_buffer_size = 1M
read_rnd_buffer_size = 1M
```

为了允许服务器执行大规模查询，需要存储大量的查询结果。控制查询规模大小的变量是 max_allowed_packet，其默认值是 1MB，我们把它更改为 16MB。

```
max_allowed_packet = 16M
```

最后，设置 log_bin 变量来启用二进制日志。万一遇到服务器崩溃，二进制日志文件将帮助我们恢复数据。

```
log_bin = /var/log/mysql/mysql-bin.log
expire_logs_days = 14
max_binlog_size = 128M
```


这里，告诉服务器每隔 14 天自动刷新一次日志文件，这意味着应该至少每两周就要对 MySQL 数据做一次备份（第 13 章内容将涉及如何自动化备份操作）。最后一行告诉服务器一旦当前的二进制日志文件达到 128MB 就创建一个新的日志文件。这样做可以避免处理庞大的日志文件，同时把刷新日志文件的操作变得简单（因为不需要在单个巨大的日志文件中提取一小部分日志）。

在 Red Hat 上，该日志目录是不存在的，所以需要创建它并且确保 MySQL 用户和组对该目录有写权限。

```
$ sudo mkdir /var/log/mysql
$ sudo chown mysql:mysql /var/log/mysql
```

现在，已经完成了 MySQL 服务器的基本设置，因此可以用“`sudo service mysqld start`”命令（Red Hat）或者“`sudo invoke-rc.d mysql start`”命令（Ubuntu）启动它。为了确认 MySQL 服务器运行良好并且创建了新的 InnoDB 事务日志文件，可以检查一下这些日志文件是否存在。在 Red Hat 上，MySQL 往文件 `/var/log/mysqld.log` 中写日志；而在 Ubuntu 上它用的是 `/var/log/syslog` 文件。

注意，我们还没有为了追求高性能而去调整 MySQL 服务器，只是修改了基本配置来提供更好的数据完整性，以及比一般情况下稍微好一点的性能。如果需要非常高的性能或者一些诸如多服务器之间数据复制这样的高级功能，建议阅读 Baron Schwartz 等人写的《High Performance MySQL, Second Edition》（O’ Reilly Media, Inc., 2008）。

11.2.4 基本的 MySQL 管理

正如读者所见的，MySQL 拥有一个内置的用户和密码列表，这意味着需要了解如何管理 MySQL 的用户，因为不希望所有的应用程序都以 root 用户连接 MySQL 服务器。我们将通过 MySQL 的命令行客户端演示如何创建和删除数据库和用户。

1. 数据库

在 MySQL 中创建数据库是简单的。通过命令行工具连接到 MySQL 服务器并且提交 `CREATE DATABASE` 语句，如列表 11-7 所示。需要把数据库名作为参数提供给 `CREATE DATABASE` 语句（注意在 SQL 语句中使用大写字母只是为了描述的清晰性，如果读者使用的是小写字母，他们依然能正确执行）。

列表 11-7 MySQL 中创建新数据库

```
$ mysql -u root -p
Enter password:
...
mysql> CREATE DATABASE 'mydb';
Query OK, 1 row affected (0.02 sec)

mysql> USE mydb;
Database changed
```



```
mysql> SHOW TABLES;
Empty set (0.00 sec)
```

在列表 11-7 中，我们创建了一个名叫 `mydb` 的数据库，然后切换到该数据库，并且检查它是否含有任何数据表。注意使用反引号引用数据库名，虽然没有明确要求使用反引号，但是某些时候数据库名、表名以及列名可能含有连字号这样的预留字符（如果不使用反引号，SQL 语句的执行可能出错）。举个例子，如果希望创建一个名为 `my-db` 的数据库，需要使用反引号，否则 MySQL 把 `my-db` 解释成从 `my` 列中的值减去 `db` 列中的值。因为这两个列都不存在，所以将产生错误，如列表 11-8 所示。

列表 11-8 合适的引号的重要性

```
mysql> CREATE DATABASE my-db;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to use
near '-db' at line 1

mysql> CREATE DATABASE 'my-db';
Query OK, 1 row affected (0.00 sec)
```

有了引号，就能用指定的名称创建数据库。

■提示：当给数据库和表命名时，通常最好使用字母、数字和下划线这些字符。即使这样，使用反引号依然是个良好的习惯。

然而我们并不需要这个数据库，所以把它删除。用 `DROP DATABASE` 语句删除它。

```
mysql > DROP DATABASE 'my-db';
Query OK, 0 rows affected (0.11 sec)
```

■注：不能撤销 `DROP DATABASE` 命令，即使在事务里面运行它然后再回滚，也不能恢复删除的数据库。所以在删除数据之前确保创建了备份。

2. 用户和权限

权限是通过 `GRANT` 语句管理的。`GRANT` 语句接受一组参数。这些参数定义了一组允许给定主机的用户执行指定对象的操作。实际情况中，通常只创建一个允许在单个数据库上执行所有操作的用户。

这就意味着每个应用程序都使用自己的数据库，并具有自己的 MySQL 登录用户。如果一个应用程序本身含有缺陷，比如允许访问数据库服务器，那么仅有该应用程序所使用的数据是危险的。

以 `root` 用户连接 MySQL 服务器，然后创建一个名为 `jsmith` 的用户——正如主机账户一样。让 `jsmith` 能访问所有数据库和表，并且能创建新用户，这么做就是为了不一直使用 MySQL 的 `root` 账户。

```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'jsmith'@'localhost' IDENTIFIED BY 'secret' ➡
WITH GRANT OPTION;
Query OK, 0 rows affected (0.12 sec)
```



```
mysql > \q
Bye
```

前面的代码创建一个名叫 jsmith 的用户。该用户只能用密码 “secret” 从本机连接 MySQL 服务器。ALL 关键字指定该用户拥有所有权限，符号 “*.*” 来表示任意数据库中的任意表。如果用 “mydb.*” 代替 “*.*”，就可以限制 jsmith 只能访问数据库 mydb。最后，指定 GRANT OPTION 给该用户使用 GRANT 语句的权限。

在此以刚刚创建的用户 jsmith 登录 MySQL 服务器，并且创建一个只能访问 mydb 数据库的用户。现在不需要指定一个 MySQL 用户来连接它，因为恰好创建了一个与主机账户同名的 MySQL 用户。

```
$ mysql -p
Enter password:
...
mysql> GRANT ALL PRIVILEGES ON 'mydb'.* to 'mydb'@'localhost' identified
      by 'passwd';
Query OK, 0 rows affected (0.00 sec)
```

现在，有了一个名叫 mydb 的用户，该用户只能访问 mydb 数据库。由于目前还不需要该用户，所以我们将通过删除用户 mydb 来展示如何删除用户。

```
mysql> DROP USER 'mydb'@'localhost';
Query OK, 0 rows affected (0.00 sec)
```

因为不需要 mydb 数据库，把它也删除。

```
mysql> DROP DATABASE 'mydb';
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> \q
Bye
```

讲解 SQL 和 MySQL 的管理技巧超出了本书的范围，但是对于许多基于 MySQL 的 Web 应用程序来说，有了这些基本技巧，我们就能按照其安装说明来创建它们。有许多网站专门教导 MySQL 技巧，而且其在线手册也是非常好的信息源，如下所示。

<http://dev.mysql.com/doc/refman/5.0/en/index.html>

<http://dev.mysql.com/doc/refman/5.0/en/grant.html>

<http://dev.mysql.com/doc/refman/5.0/en/drop-user.html>

■提示：学习 MySQL 技巧非常有用的一本书是 W. Jason Gilmore 所著的《PHP 与 MySQL 程序设计：第三版》。

11.3 安装网络站点

既然已经配置了 Web 和 MySQL 服务器，现在就可以安装一些 Web 应用程序来增强网络

交流。在本节，我们会展示如何安装一些 Web 应用程序，但不会展示如何使用这些 Web 应用程序，因为它们大部分都带有优秀的说明文档和专门的社区支持。

我们先创建一个名叫 `www` 的用户组，然后向该组添加需要修改 Web 站点配置的用户，另外把自己也添加到该组。可以通过 `-k` 参数更改该用户组的默认 `umask`。

```
$ sudo groupadd -K UMASK=0002 www
$ sudo usermod -G www jsmith
```

一旦我们注销再登录系统，组隶属关系的修改将会生效。

然后，更改目录 `/srv/www/www.example.com` 和它所包含的所有子目录的所有权和访问权限，让 `www` 用户组对它们有全部的访问权限。

```
$ sudo chgrp -R www /srv/www/www.example.com
$ sudo chmod u+rwx,g+srwx,o+rx /srv/www/www.example.com
$ sudo chmod u+rwx,g+srwx,o+rx /srv/www/www.example.com/html
```

可以用一个精确的八进制形式 `2775` 代替表示全额权限的字符串 (`u+rwx,g+srwx,o+rx`)。在 Red Hat 上，还需要告诉 SELinux 这个新的目录被用作 Web 目录。

```
$ sudo chcon -t httpd_sys_content_t /srv/www/www.example.com/html
```

如果遗漏这一步，SELinux 会阻止 Apache 访问该目录中的任何文件。

11.3.1 网络交流

当然，我们希望不需要自己花力气去学习 HTML 也能让企业拥有一个网络交流服务。为此，安装一个内容管理系统(CMS)做我们的网站。

CMS 在提供一个框架的同时，还允许用户集中精力创建内容和美化页面。CMS 框架可以减少页面频繁修改，把 Web 内容和图形设计相分离，管理用户和员工的访问权限。

我们将在虚拟主机 `www.example.com` 上安装一个叫做 Drupal 的 CMS。如果读者偏好其他 CMS 系统——比如 Joomla(<http://joomla.org>)或者 eZ Publish(<http://ez.no>)——当然也可以安装它们。选择 Drupal 是因为 Drupal 比较成熟、有良好的支持并且容易扩展。

可以从 <http://www.drupal.org> 上获取该软件的源码包。在书写本章节时，Drupal 的最新版本是 Drupal 6.10。现在用 `wget` 工具把它下载到主目录。

```
$ wget http://ftp.drupal.org/files/projects/drupal-6.10.tar.gz
--11:07:45-- http://ftp.drupal.org/files/projects/drupal-6.10.tar.gz
Resolving ftp.drupal.org... 64.50.236.52, 64.50.238.52
Connecting to ftp.drupal.org|64.50.236.52|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1075558 (1.0M) [application/x-gzip]
Saving to: 'drupal-6.10.tar.gz'

100%[=====>] 1,075,558 382K/s in 2.7s

11:07:48 (382 KB/s) - 'drupal-6.10.tar.gz' saved [1075558/1075558]
```


然后，解压源码包。可以在根目录解压它，然后把需要的文件和目录移动到 Web 根目录，或者直接把它解压到 Web 根目录。我们使用后一种方法，使用 `tar` 命令从压缩包中提取第一个目录部分，并且使用 `-C` 选项指定一个目标目录。

```
$ tar -xz --strip-components=1 -C /srv/www/www.example.com/html -f drupal-6.10.tar.gz
```

提示：为了发现 `tar` 压缩包里含有哪些目录，可以使用 `-t` 和 `-v` 选项来显示 `tar` 包中的文件列表（比如“`tar -tvzf drupal-6.10.tar.gz`”）。

既然已经安置好网站的内容，现在可以在浏览器中访问 `http://example.com/`。这会启动 Drupal 基于网络的安装过程，如图 11-12 所示。

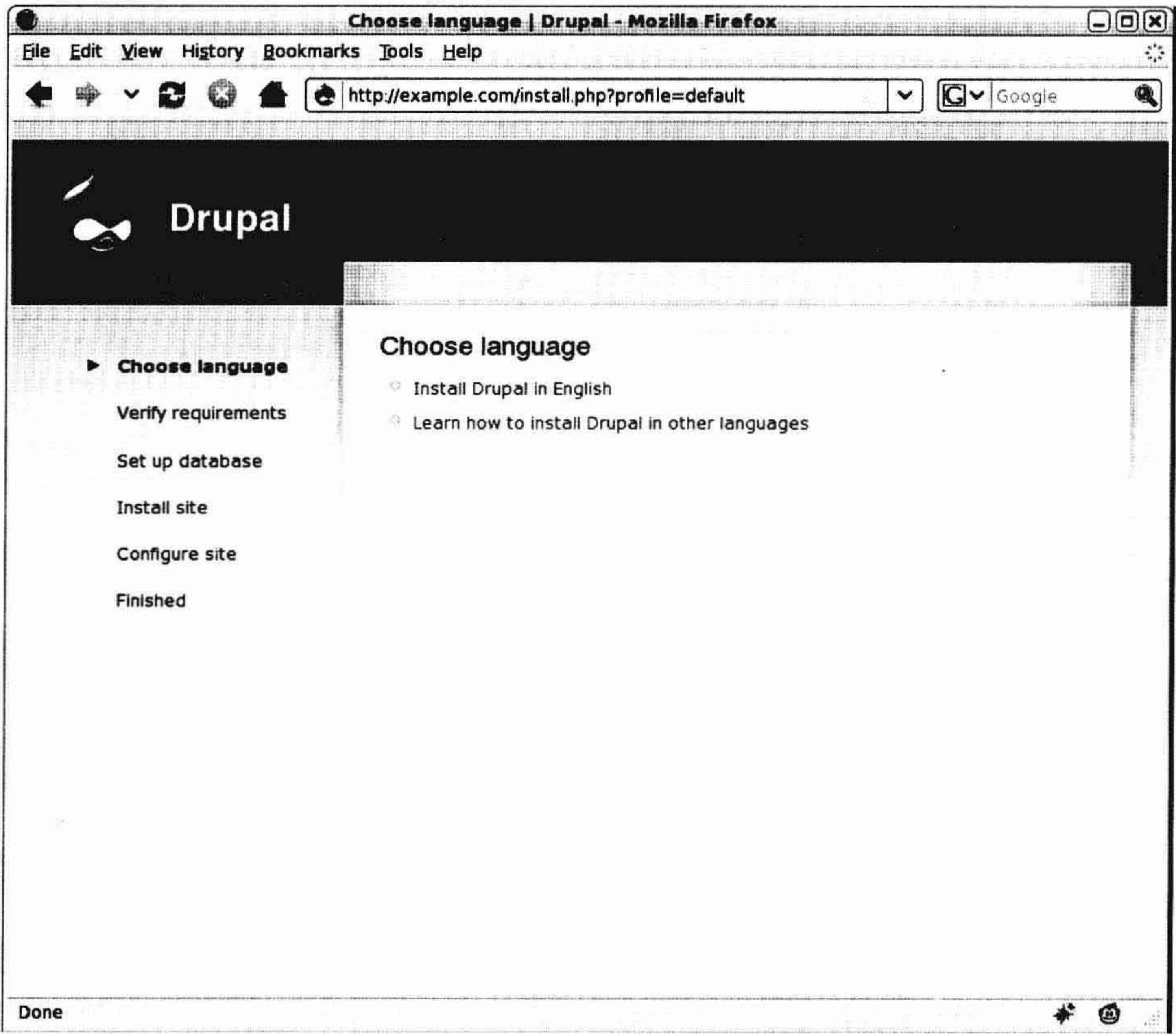


图 11-12 安装 Drupal

单击“Install Drupal in English”继续安装。下一页告诉我们安装程序需要我们在硬盘上创建一个文件和一个目录。安装程序会向该文件中写入 Drupal 的设置信息，而该目录被 Drupal 用来存储文件，如图 11-13 所示。

创建所要求的文件和目录，并给它们适当的所有权和访问权限。

```
$ cd /srv/www/www.example.com/html
$ cp sites/default/default.settings.php sites/default/settings.php
$ mkdir sites/default/files
```

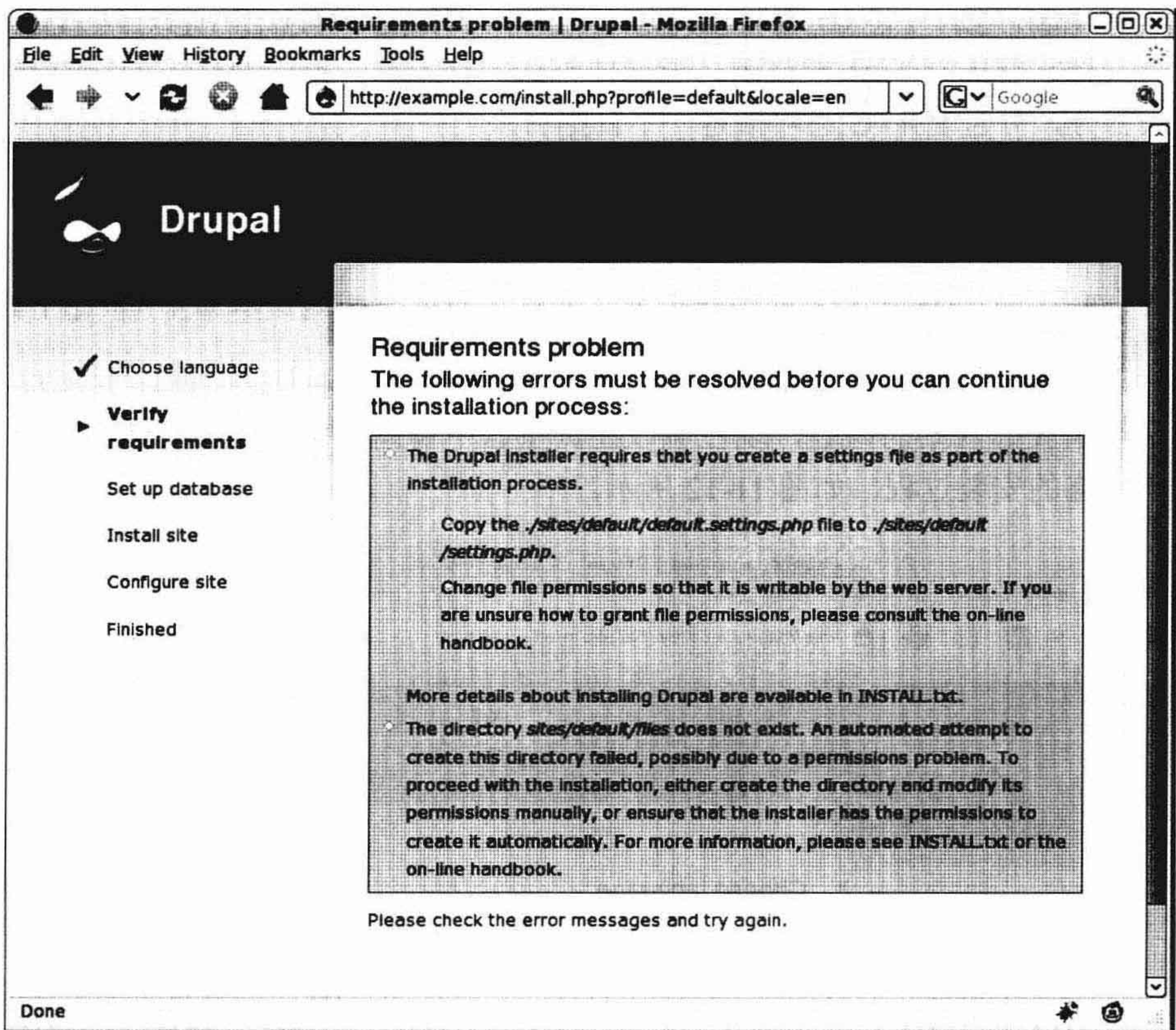



图 11-13 提示我们要创建一个可写的文件

可以用两种方式管理这些文件或目录的写权限，使该文件和目录对“other”用户可写，或者使它们对 www 用户组可写，但是把它们的主改为 Web 服务器赖以运行的用户。第二种方法能防止其他用户重写 Drupal 的配置文件以及在 Drupal 的文件目录下存储数据。

在 Red Hat 上，Web 服务器以 Apache 用户身份运行；在 Ubuntu 上，它以 www-data 用户身份运行。所以需要按照需求改变该文件和目录的主。

```
$ sudo chown apache sites/default/settings.php
$ sudo chown apache sites/default/files
```

既然已经更改了这些权限，就可以在 Web 浏览器中重新加载这个页面。现在，我们被带到一个需要输入 MySQL 信息的页面。首先，创建一个数据库和一个用户。这里，最好遵循一个惯例，也就是在给数据库命名时，最好与该应用程序的名称一样，并且也用同一个名称创建数据库用户。只要选择一个安全的密码，就不会有什么问题。

```
$ mysql -p
Enter password:
...
mysql> CREATE DATABASE 'drupal';
Query OK, 1 row affected (0.04 sec)

mysql> GRANT ALL PRIVILEGES ON 'drupal'.* to 'drupal'@'localhost' IDENTIFIED BY 'Eishao4h';
```


Query OK, 0 rows affected (0.20 sec)

mysql> \q
Bye

现在，可以向 Drupal 安装程序输入这些详细信息，如图 11-14 所示。

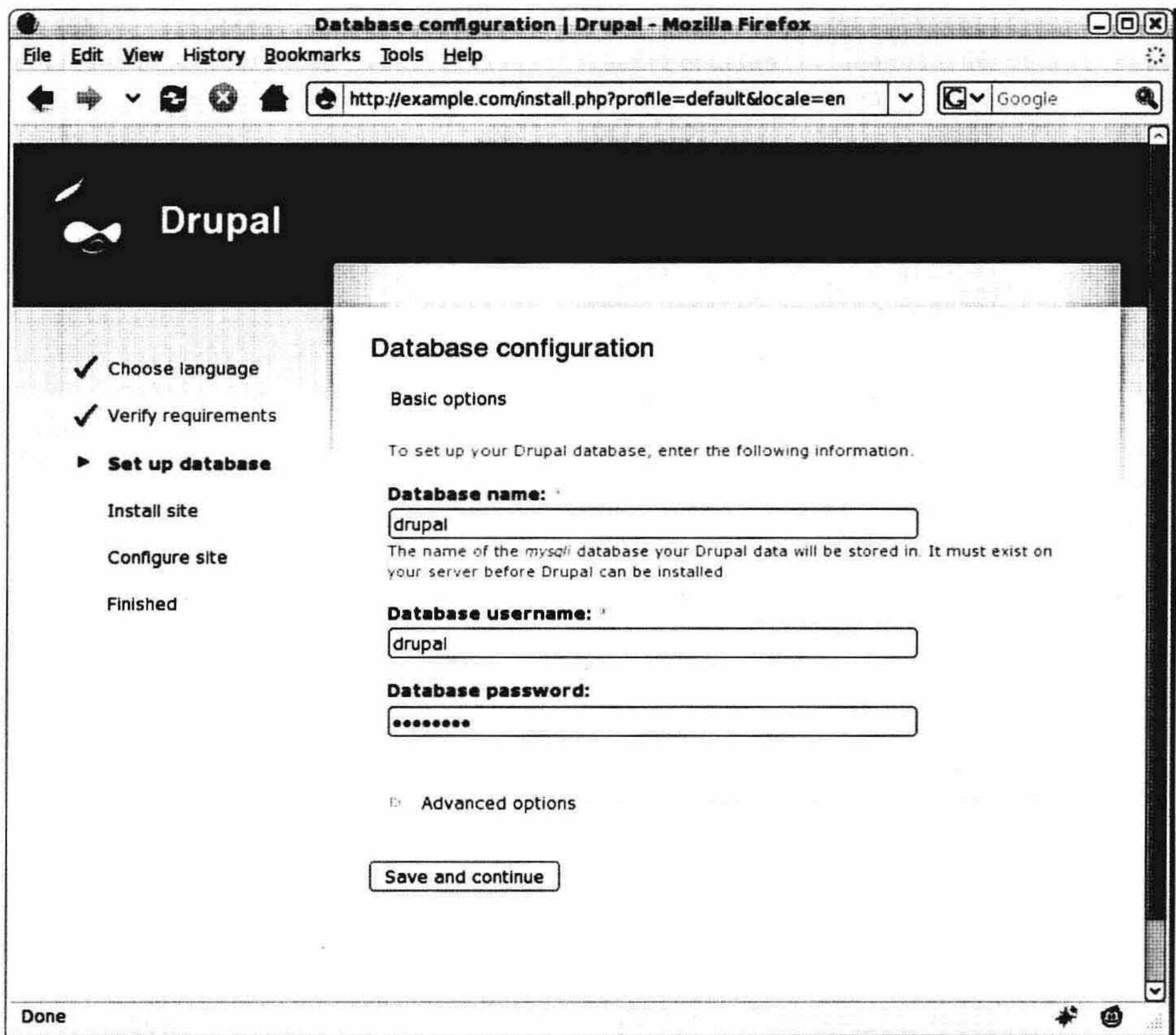


图 11-14 输入数据详细资料

单击“Save and continue”按钮开始安装过程。安装程序将向数据库填充所需的表和数据。这时，一个进度条通知我们安装的进程，如图 11-15 所示。

当安装完成时，我们被带到网站基本配置页面。在这里设置站点名称，创建管理员账号，然后单击“Save and continue”按钮。Drupal 会完成配置并通知我们访问新站点。

注：当安装完成时，别忘了把文件 `sites/default.settings.php` 的属主改回原来的属主，那样 Apache 就不能修改它了。

现在可以从首页开始管理站点了，如图 11-16 所示。
大量有关 Drupal 的信息可在 <http://drupal.org> 上获得。
用户手册：<http://drupal.org/getting-started>
主题：<http://drupal.org/project/themes>
模块：<http://drupal.org/project/modules>

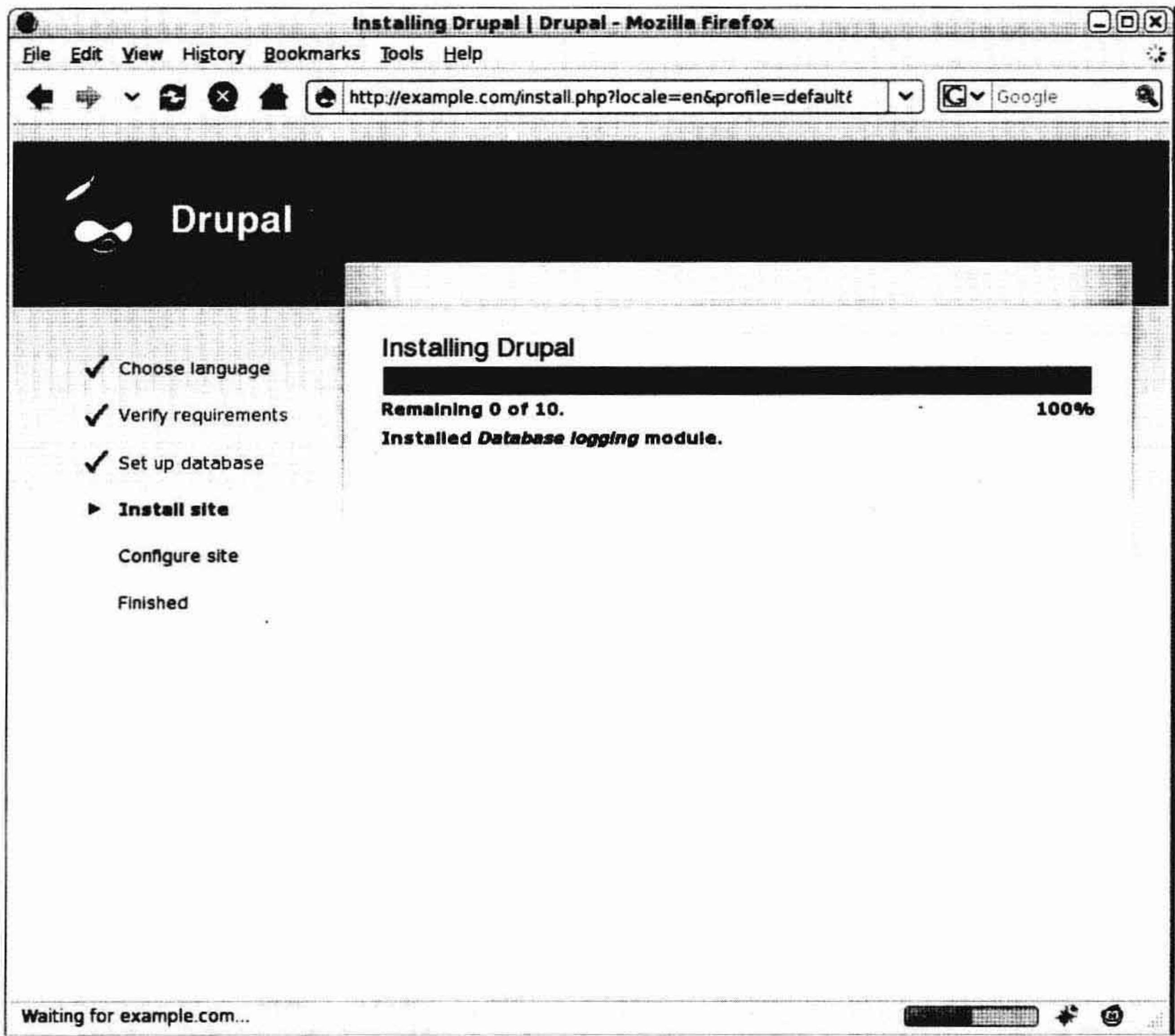


图 11-15 安装进程

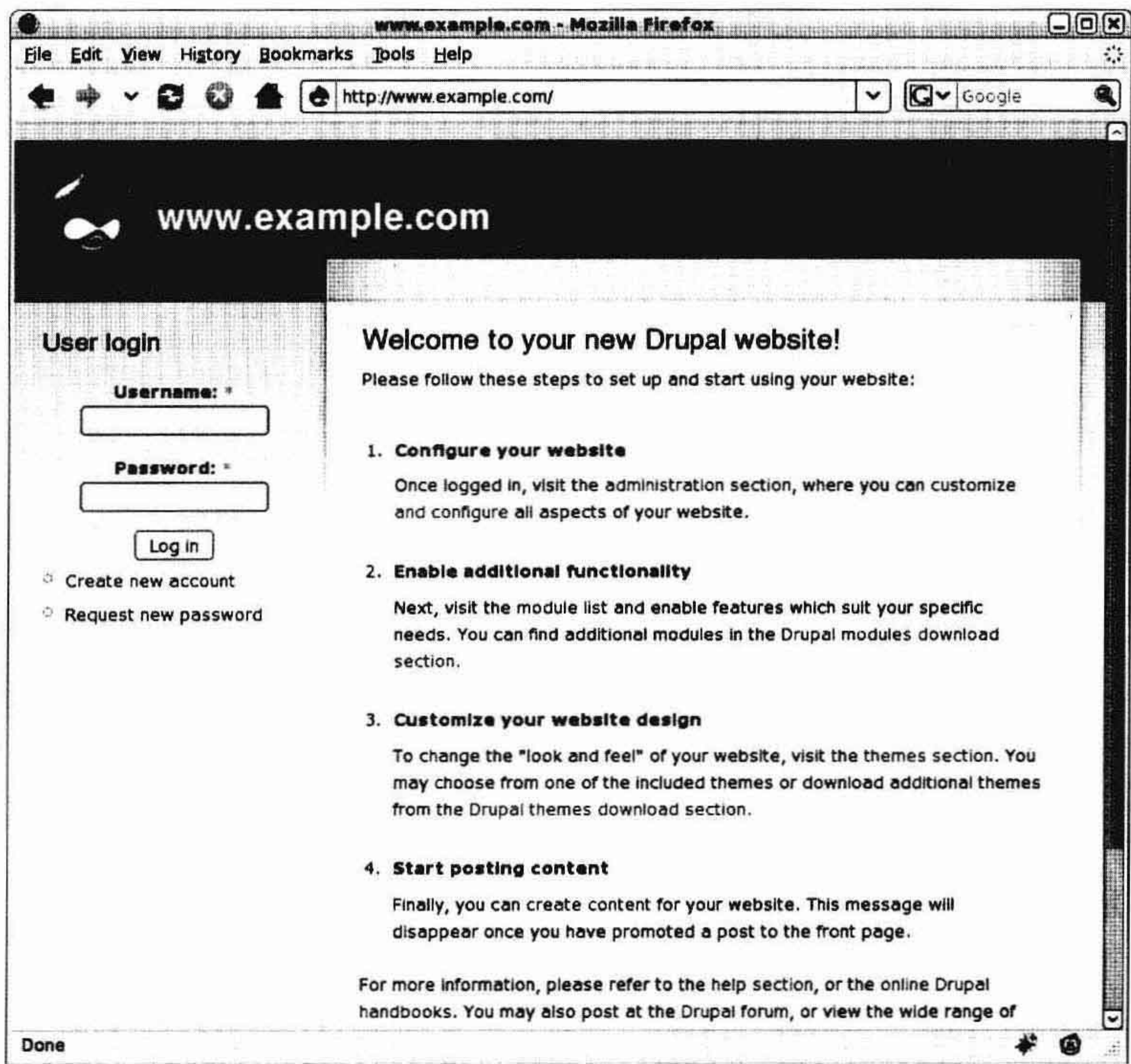


图 11-16 新的在线交流系统

11.3.2 · webmail

为了给用户远程访问 E-mail 服务，将在 Web 虚拟主机上安装一个 webmail 软件包。同时把该主机称作 `webmail.example.com`。首先，需要给服务器添加一个 IP 地址，并且在 DNS 服务器里为该地址新建一个条目。

给网络中的主机分配地址时，最好给虚拟网络站点预留一些地址。这里我们将使用地址 `192.168.0.220`。

■注：第 6 章解释如何向网络接口添加额外的地址；第 9 章讨论如何添加 DNS 主机名和别名。

接下来，需要在 Apache 中建立一个虚拟主机。该虚拟主机根目录为 `/srv/www/webmail.example.com/html`，寻址方式为以 IP 寻址。由于我们想用 SSL 来保护用户隐私，因此没有选择以名称寻址的虚拟主机。为此，还要创建一个证书供 Apache 使用。这里，用 `openssl` 就可以了，这和为 mail 服务器创建证书时所采用的方法完全一样。

```
$ openssl req -new -newkey rsa:4096 -nodes -keyout webmail.example.com.key ➡
-out webmail.example.com.req
```

■注：不要忘了使用 CA 分配服务器证书，就像第 10 章讨论的一样。

最好把所有虚拟主机相关的证书文件存储在一个地方，为此创建一个名为 `/srv/www/webmail.example.com/ssl` 的目录，并且把证书和所创建的密钥移动到该目录。用 `chmod` 命令设置适当的权限，确保 Apache 用户账户可以读取这些文件，如果有必要，在 Red Hat 上还需要用 `chcon` 命令设置正确的 SELinux 安全环境。

```
$ sudo chmod 0640 /srv/www/webmail.example.com/ssl/*
$ sudo chcon-R system_u:object_r:cert_t /srv/www/webmail.example.com/ssl
```

1. Red Hat

在 Red Hat 上，首先需要为 Apache 安装 SSL 支持。Red Hat 上 Apache 的 SSL 支持是由 `mod_ssl` 软件包提供的，可以用“`sudo yum install mod_ssl`”命令安装它。接着，需要禁用已经启用模块 `mod_ssl` 的默认的 SSL 虚拟主机，否则该主机将影响我们的配置。

在文本编辑器里，打开文件 `/etc/httpd/conf.d/ssl.conf`，并且注释掉位于 81 行和 228 行之间的 `VirtualHost` 指令，即在行首添加一个散列符号 `#`。

```
# <VirtualHost _default_:443>
# </VirtualHost>
```

保存这些修改，重启 Apache 并且通过菜单“系统”>“管理”>“服务器设置”>“HTTP”启动 Apache 的配置工具。

接下来，在该配置工具中选择“虚拟主机”标签页，并且单击“添加”按钮添加一个新的虚拟主机。给虚拟主机取一个说明性的名称，并且输入正确的根目录和 e-mail 地址（见

图 11-17)。

在“Host Information”下面，选择“基于 IP 的虚拟主机”，并且输入使该主机有效的 IP 地址和端口号以及满意的主机名称。接着转到“页码选项”标签页，添加 index.php 到“Directory Page Search List”。这将确保当用户访问该网站时，Apache 会处理和显示文件 index.php(如果它存在)。

最后，切换到“SSL”标签页，勾选“启用 SSL 支持”复选框。这将启用下面的文本域，告诉服务器使用哪个证书和密钥。

在“证书文件”和“证书密钥文件”文本域中输入它们的全路径，如图 11-18 所示。

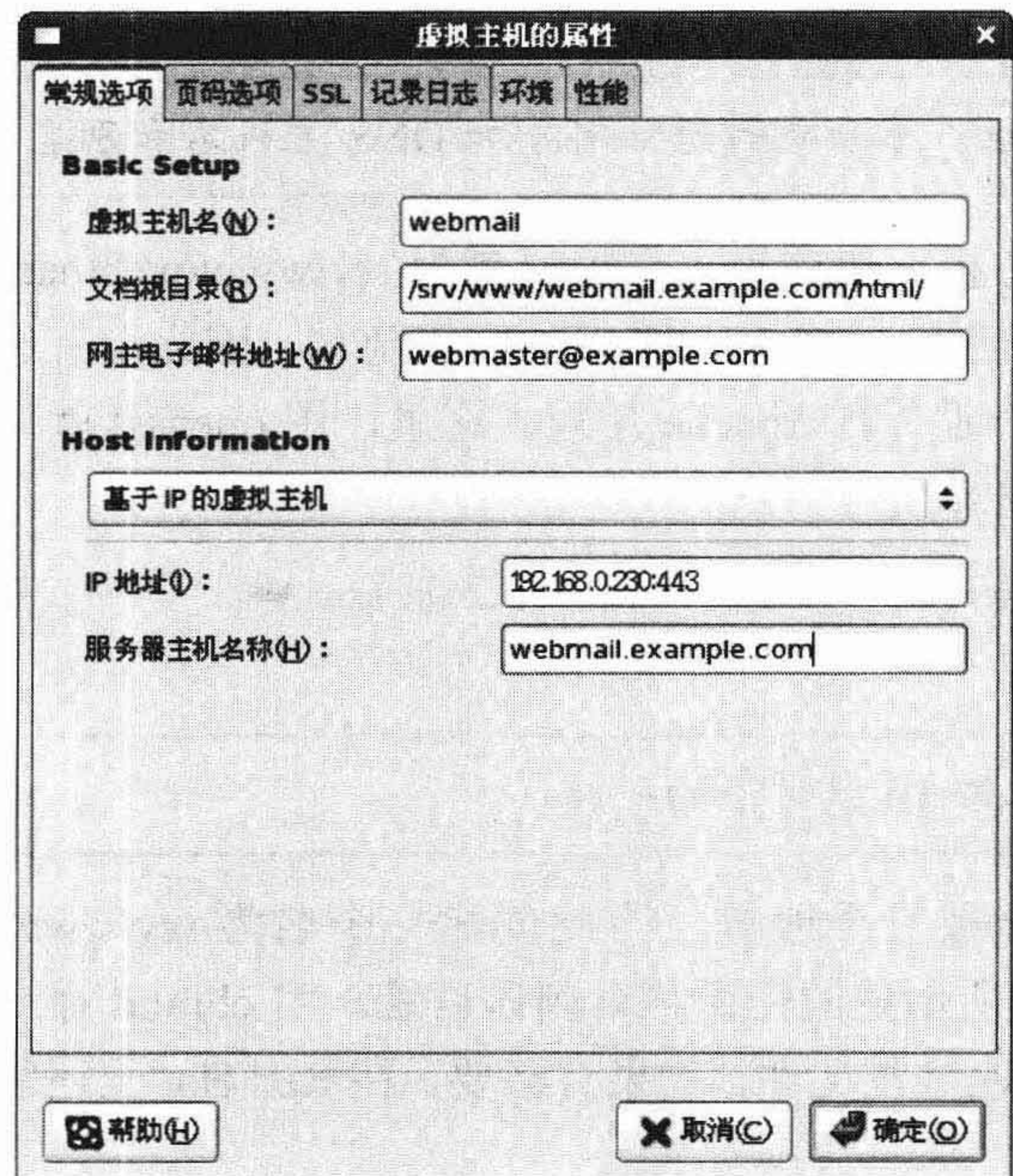


图 11-17 在 Red Hat 上创建以 IP 寻址的虚拟主机

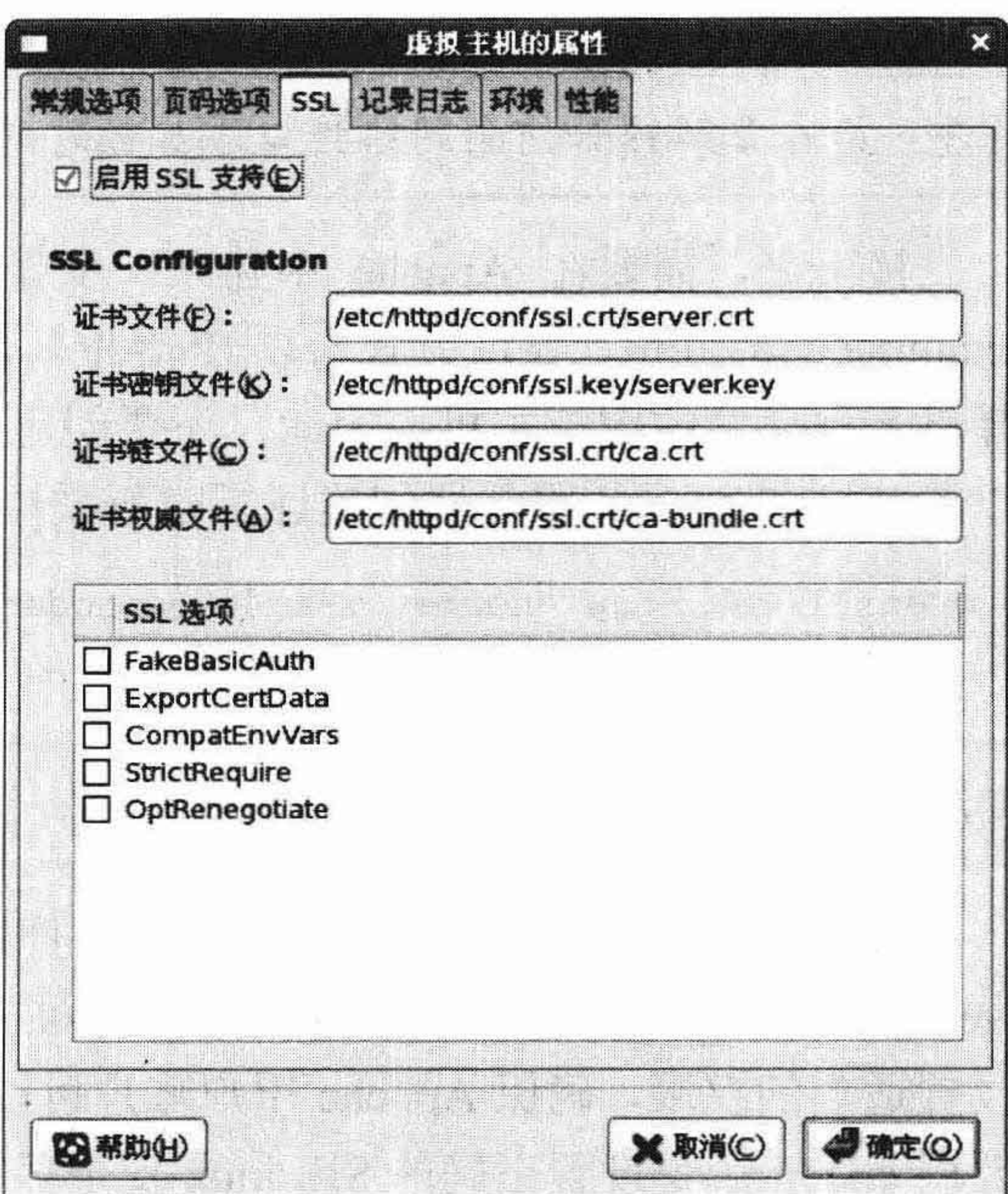


图 11-18 设置 SSL 属性

如果从一些供应方手里购买商业的证书（比如 GoDaddy 和 InstantSSL），那么还需要安装一个证书链文件。证书链文件包含了提供方签署的证书，浏览器需要用它来确定证书上的签名是有效的。

自己签署的证书就不必要带有这样一个证书链文件，但是如果不填写“证书链文件”这个域，Apache 将启动失败。为了避免这个问题，可以在“证书链文件”文本域中输入自己签署的证书文件的路径，同时在“证书权威文件”域中输入它。

现在虚拟主机已经配置好了，可以接受这些设定的值，并且关闭 HTTP 工具，保存更改。最后，让防火墙允许 TCP 443 端口上的通信，让用户具备穿越防火墙访问 SSL 网站的权限。

2. Ubuntu

在 Ubuntu 上，Apache 的 SSL 模块是默认安装的，但是在使用它之前需要通过“sudo a2enmod ssl”启用它。接着要告诉 Apache 我们想在地址 192.168.0.220 的 SSL 端口上监听请求。

正如之前看到的，可以在文件/etc/apache2/ports.conf 中进行配置。为了确保 SSL 模块被载入时 Apache 只监听该端口，需要把 Listen 指令设置在一个 IfModule 语句里。同时，为了确保

Apache 不会在所有地址上监听 443 端口，在端口前面加上该地址，如列表 11-9 所示。

列表 11-9 有条件地监听 SSL 端口

```
<IfModule mod_ssl.c>
    Listen 192.168.0.200:443
</IfModule>
```

接下来，需要创建虚拟主机文件(virtual host file)，创建该文件为/etc/apache2/sitesavailable/webmail.example.com.conf。在该文件中，把配置包含在一个 IfModule 语句里，这样即使 SSL 模块没有成功载入，SSL 指令也不会导致 Apache 启动失败。

设定虚拟主机只在地址 192.168.0.220 的端口 443 上监听，并且设置 ServerName、ServerAdmin 和 DocumentRoot。同时还需要在根目录中设置一些默认选项，如列表 11-10 所示。

列表 11-10 安全的虚拟主机配置

```
<IfModule mod_ssl.c>
<VirtualHost 192.168.0.220:443>
    ServerName webmail.example.com
    ServerAdmin webmaster@example.com
    DocumentRoot /srv/www/webmail.example.com/html
    <Directory /srv/www/webmail.example.com/html>
        Options FollowSymLinks Includes IncludesNOEXEC SymLinksIfOwnerMatch
        AllowOverride All
    </Directory>
    SSLEngine on
    SSLCertificateFile /srv/www/webmail.example.com/ssl/webmail.example.com.cert
    SSLCertificateKeyFile /srv/www/webmail.example.com/ssl/webmail.example.com.key
    SSLCACertificateFile /etc/CA/cacert.pem
</VirtualHost>
</IfModule>
```

为了保证该站点的所有通信都是加密的，需要为该虚拟主机打开 SSL 支持。可以用“SSLEngine on”指令开启 SSL 支持，因为“SSLEngine on”指令被包含在一个 VirtualHost 区块里，所以它只为该指定的虚拟主机开启 SSL 支持。

最后，需要告诉 SSL 引擎使用哪个主机证书、哪个主机密钥以及哪个 CA 证书。之后就可以启用该虚拟主机（用“sudo a2ensite webmail.example.com”命令）和重启 Apache。

11.3.3 配置 SquirrelMail

现在，我们将展示如何在安全的、新建的虚拟主机上创建 SquirrelMail Web 应用程序。SquirrelMail 只是众多 webmail 应用程序中的一员。读者可能使用过其他的 webmail，比如 RoundCube(<http://roundcube.net>)和 Horde Project 的 IMP(<http://horde.org/imp>)。

可以从 <http://squirrelmail.org/> 上获得 SquirrelMail，在书写本章节时，SquirrelMail 的最新版本是 1.4.7。

我们把 SquirrelMail 源码包下载到主目录，并将它释放到适当的目录。

```
$ tar -xz --strip-components=1 -C /srv/www/webmail.example.com/html
-f squirrelmail-1.4.17.tar.gz
```


在浏览器中访问新建的虚拟主机时会遇到错误，因为 SquirrelMail 并没有提供基于 Web 的配置过程。然而，该软件包带有的 README 和 INSTALL 两个文件会告诉我们如何安装。

```
$ cd /srv/www/webmail.example.com/html
$ less README
$ less INSTALL
```

可以按照快速安装向导进行安装。首先，跳过已经完成的前 3 步。接着，为站点的配置文件和附件创建目录。因为我们的 Web 目录树是虚拟主机根目录中的子目录，所以可以在目录 /srv/www/webmail.example.com/ 下面创建目录，而不是按照安装文档的建议，在目录 /var 下面创建目录。这样做在做备份的时候会有好处，因为所有与虚拟主机相关的数据都放置在同一个地方，所以不太可能忘记备份某些数据。

在虚拟主机主目录下创建目录 data 和 attach，并且按照 INSTALL 文件的说明更改它们的所有权，以便 Apache 用户可以对它们有写权限。

```
$ cd /srv/www/webmail.example.com
$ sudo mkdir data attach
$ sudo chown apache:www data
$ sudo chgrp apache attach
$ sudo chmod 0730 attach
```

在 Red Hat 上，还需要改变在这些目录上的 SELinux 标签。

```
$ sudo chcon -t httpd_sys_content_t data attach
```

接着，继续下一步，运行配置脚本。

```
$ cd /srv/www/webmail.example.com/html
$ ./config/conf.pl
```

现在，可以使用 SquirrelMail 配置工具了，如图 11-19 所示。

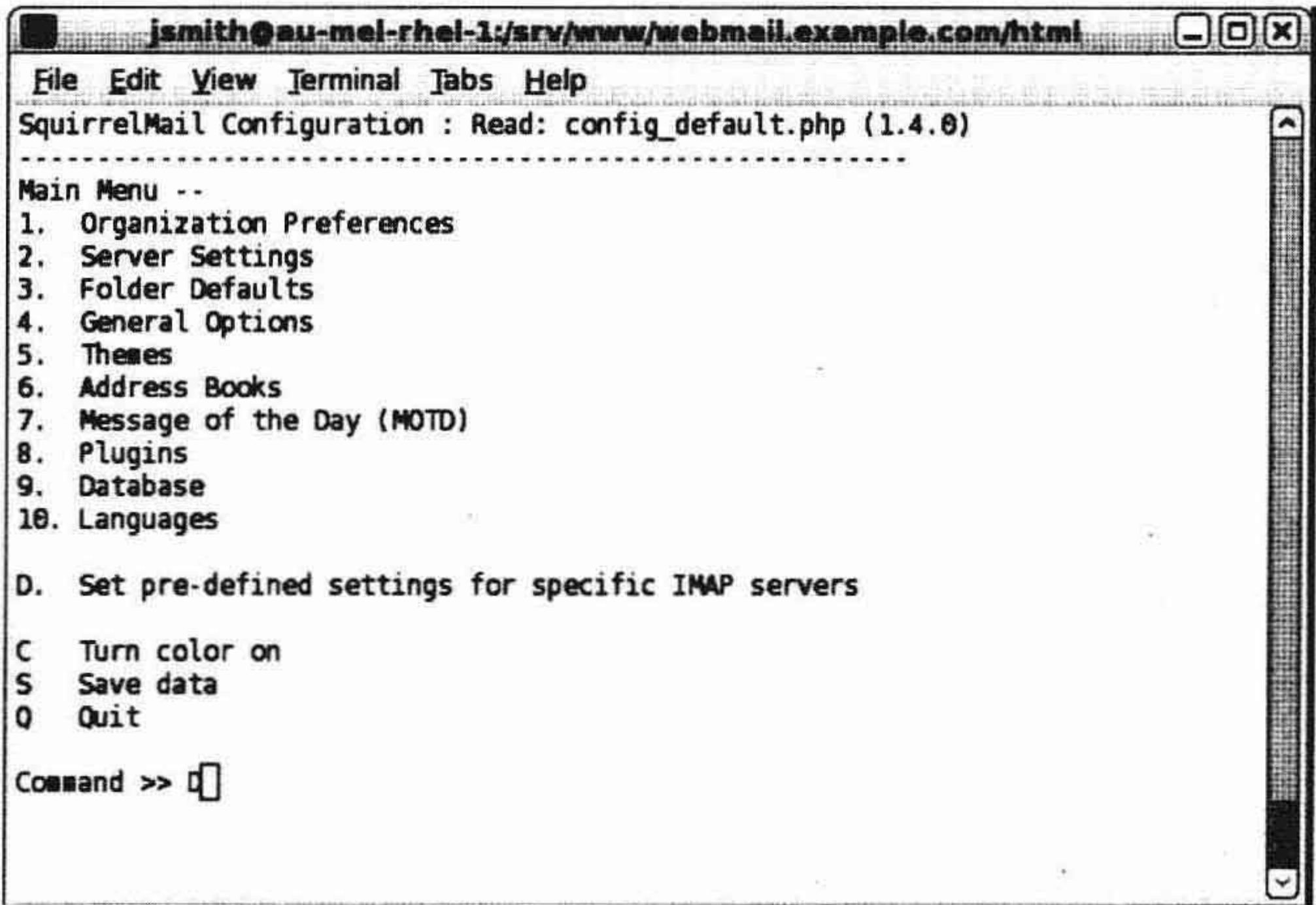


图 11-19 SquirrelMail 配置

输入 “D” 并且按回车键。在下一个界面里，需要为 IMAP 服务器(在我们的案例中是 dovecot)载入默认选项。键入服务器名并再次按回车键，可以看到刚刚设置的详细的 IMAP 选项（见图 11-20）。

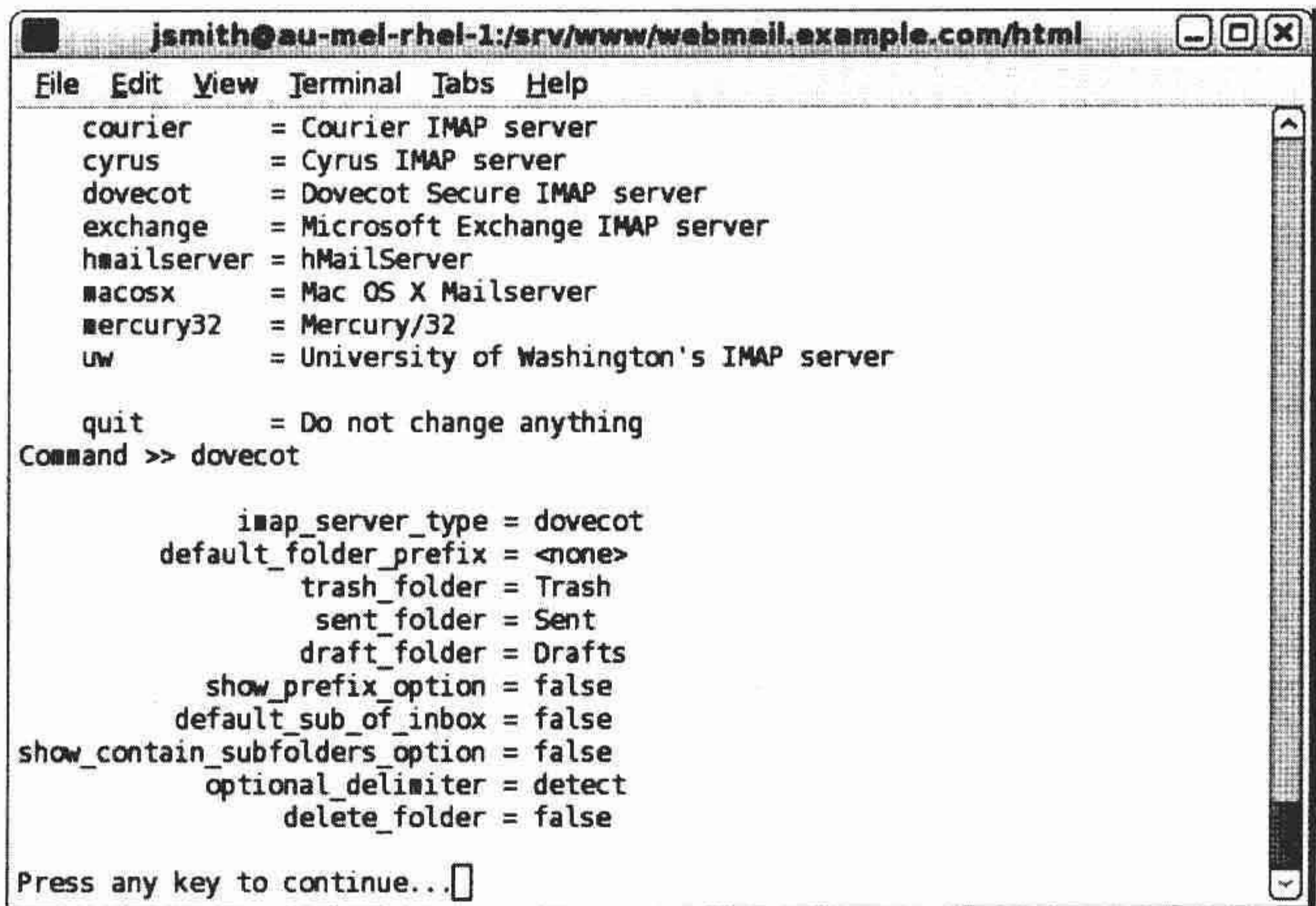


图 11-20 选择 IMAP 服务器软件

再次按回车键，接受这些设置并返回主菜单。

下一步，更改邮件服务器的设置。选择第二个菜单项，因为我们的 Web 主机同样也是邮件主机，在这不需要做任何修改。然而，如果 Web 和邮件使用不同的主机，还要更改 IMAP 和 SMTP 两者的主机设置。输入“r”并且按回车键返回主菜单。

还需设置 SquirrelMail 的数据目录和附件目录，可以用“常规选项”设置这些目录。按下 4 键并且按回车键。然后，按下 1 键设置数据目录，接着按回车键。现在可以输入之前创建的目录的全路径，如图 11-21 所示。完成之后再按回车键。

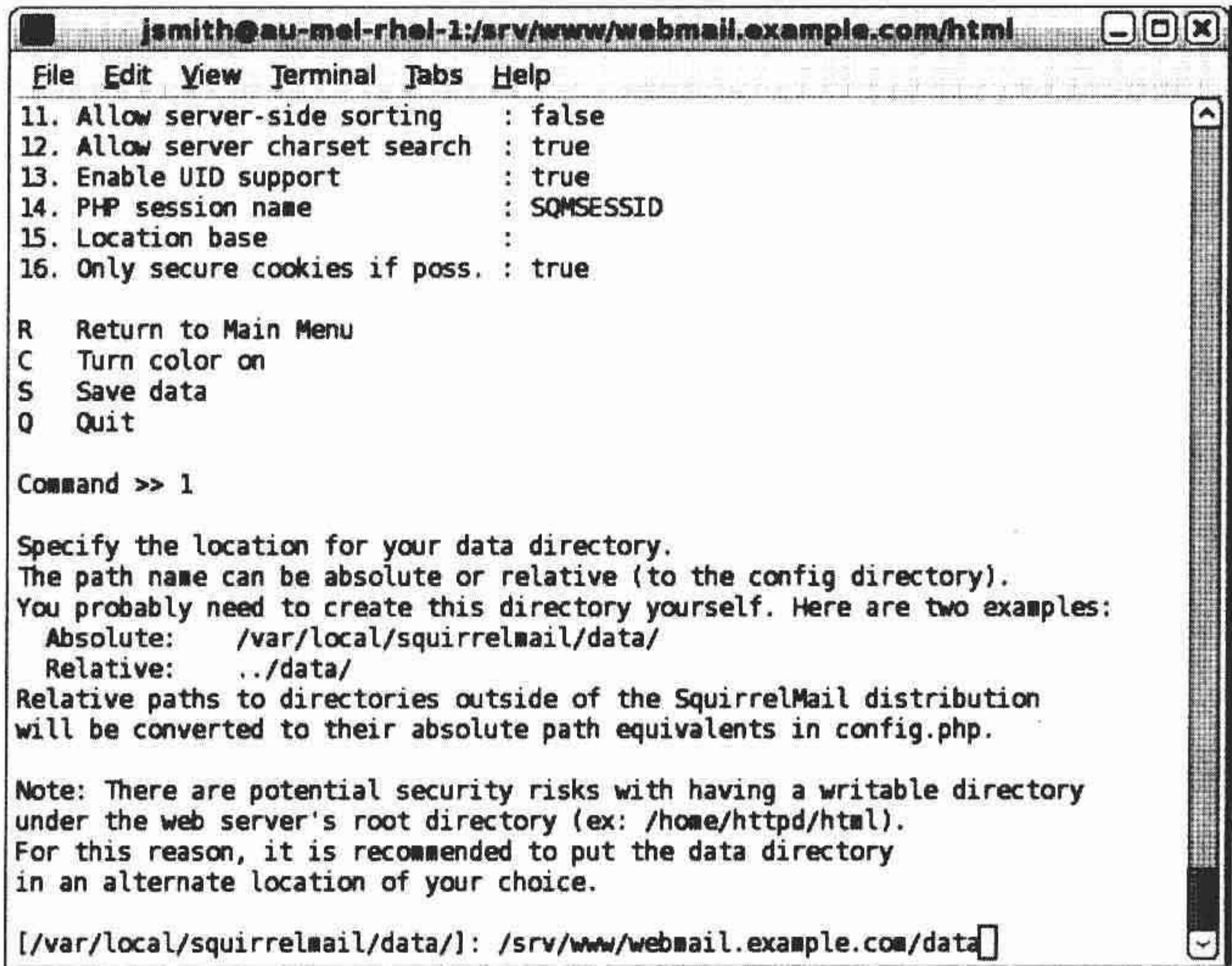


图 11-21 设置数据目录

选择选项 2，设置附件目录为/srv/www/webmail.example.com/attach。

基本配置已经完成。按下“r”键返回主菜单，然后先后按下“s”和回车键保存配置，该工具将创建配置文件 config.php，并把配置写入该文件。最后按下“q”键关闭该工具。

刷新 Web 浏览器，将看到 SquirrelMail 的登录界面，如图 11-22 所示。



图 11-22 可以从任何地方登录 SquirrelMail

11.3.4 其他 Web 应用程序

我们已经展示了如何把两个 Web 应用程序安装到它们各自的虚拟主机上，这应该使读者对安装 Web 应用程序的过程有了基本的认识。我们不可能在这包含所有 Web 应用程序的安装向导，但是可以向读者建议一些有用的工具。

phpMyAdmin 工具为 MySQL 提供了一个基于 Web 的界面。它允许我们创建数据库和表、导入和导出数据以及管理用户。可以从 <http://www.phpmyadmin.net/> 站点下载它。

MediaWiki 是运行 Wikipedia（许多使用 MediaWiki 的站点之一）的一个协作工具。可以从 <http://www.mediawiki.org/> 站点下载它。

Moodle 是个课程和学习管理系统。许多学校用它来给学生提供网络学习平台。可以从 <http://www.moodle.org/> 站点下载它。

SugarCRM 是可以帮助我们和组织的销售人员管理客户和客户关系的客户关系管理（customer relationship management, CRM）软件。可以从 <http://www.sugarcrm.com/> 站点下载它。

11.4 Squid Cache

当公司中多个用户同时浏览 Web 时，就可能占用大量的带宽。为了减少 Web 浏览对其

他网络用户的影响，可以安装一个缓存服务器或者代理服务器。

Linux 上最常用的缓存服务器是 Squid。如果浏览器配置使用该缓存，Squid 将保存从网上检索的任何文件的本地副本，所以浏览器下一次访问同一个文件时，可以从本地 Squid 缓存中获取它。

另一种使用 Squid 的方式是把它作为反向代理 (reverse proxy)。反向代理位于一个或者多个 Web 服务器之前，它通常用于那些拥有大量静态页面的网站。把不变的数据缓存到反向代理中，可以减轻目标 Web 服务器的负担。这种代理也称作网页加速器 (Web accelerator)。使用 Squid 作为网页加速器的一个样例是 Flickr(<http://www.flickr.com>)，它是个大型网站。

11.4.1 配置

Squid 是由 squid 包提供的。在 Red Hat 上，用 “`sudo yum install squid`” 命令安装它；而在 Ubuntu 上用 “`sudo aptitude install squid`” 命令安装它。两个发行版本都使用相同的配置文件，它就是 `/etc/squid/squid.conf`。因为该文件可能含有敏感信息（比如密码），不用 `sudo` 命令甚至不能查看它。

Squid 默认的配置文件中良好的注释，所以我们不会对大部分选项做非常细致的说明。然而，我们会展示如何配置 Squid 所监听的地址和端口，以及如何让用户真的能通过缓存来访问 Web。

默认情况下，Squid 在所有网络接口的 3128 端口上监听，不过 8080 也是它常用的端口。配置文件中修改端口的指令是 `http_port`。该指令指定一个端口号，或者将一个地址和端口号相结合。如果希望 Squid 在多个地址和端口上监听，可以添加更多 `http_port` 指令。

因为要在网关的主机上建立 Squid，所以不要让 Squid 在所有接口地址上监听连接，从而使外网的用户不能访问它。添加两行 `http_port` 指令，一行为无线网络地址域，一行为有线网络地址域。

```
http_port 192.168.0.254:3128
http_port 192.168.1.254:3128
```

接着，需要告诉 Squid 允许哪些 IP 地址域连接和访问。Squid 用访问控制列表 (access control lists, ACLs) 控制连接。我们需要为每个想控制的网络地址域定义一个 ACL，然后为每个 ACL 创建控制访问的规则。Squid 的配置文件包含一些基本的 ACLs，如列表 11-11 中所包含的这些 ACLs。

列表 11-11 ACL 定义

```
acl all src 0.0.0.0/0.0.0.0
acl localhost src 127.0.0.1/255.255.255.255
acl to_localhost dst 127.0.0.0/8
```

`acl` 指令告诉 Squid 该行剩下的部分是一个 ACL 定义。接着，给 ACL 一个标签，以便随后可以引用它。然后指定所创建的 ACL 的类型。Squid 支持多种类型，但是我们只使用 `src`、`dst` 和 `port` 3 种类型。这 3 种类型分别控制是否处理源地址、目标地址和端口号。最后，分别用一个字符串定义源地址、目标地址和端口号。为了获得 ACL 类型的全部列表，可以在 Squid 网站的 <http://www.squid-cache.org/Versions/v2/2.6/cfgman/acl.html> 页面中仔细阅读关于 ACL 的章节。

在本例中，定义了一个包含所有网络地址的、叫做 `all` 的 ACL。下一个 ACL 以 `localhost` 字符串指代局域网接口上的所有通信源。第三条指令做同样的事，但只针对到本地接口的通信。

接下来给网络地址添加一些 ACLs，可以直接在定义 `to_localhost` 的那行下面添加配置。


```
acl wired src 192.168.0.0/255.255.255.0
acl wireless src 192.168.1.0/255.255.255.0
```

这里，我们已经定义了两个新的 ACLs，一个为有线网络域，另一个为无线网络域。通过为两个 ACLs 指定相同的标签，可以把它们结合在一起。而给它们不同的名称则意味着可以给它们设定不同的访问级别。

Squid 包含一个名叫 `Safe_ports` 的 ACL。`Safe_ports` 含有 Web 和 FTP 通信的常用端口。这样，就能够控制远程服务器上的哪些端口允许本地用户连接。如果要访问的某个站点使用了一个非标准的端口，可以把该端口号添加到 `Safe_ports` ACL，并且允许浏览器连接它。

```
acl Safe_ports port 80      # http
acl Safe_ports port 21     # ftp
acl Safe_ports port 443    # https
```

既然已经定义了全部所需的 ACLs，就可以为这些 ACLs 定义访问规则完成配置。Squid 使用 `http_access` 指令来规定一个给定的 ACL 是否允许使用缓存。该指令含有两个参数：一个动作和一个 ACL。这些指令依次被处理，并且当满足一条规则时，处理过程便会停止。

第一条规则防止浏览器连接没有明确地在 `Safe_ports` ACL 中列出的端口。

```
http_access deny !Safe_ports
```

接下来，允许 `localhost` ACL 的连接，并且拒绝其他所有的连接。

```
http_access allow localhost
http_access deny all
```

如果没有任何一条 `http_access` 规则被满足，最后一个看到的动作会被使用。这就是为什么总是把“`http_access deny all`”留做最后一条规则，它意味着 Squid 如果没有遇到一条允许访问的规则就拒绝访问。

我们应该在它们之间插入新规则，如列表 11-12 所示。

列表 11-12 给网络赋予访问权限

```
http_access allow localhost
http_access allow wired
http_access allow wireless
http_access deny all
```

最后，可以通过 `cache_dir` 指令更改 Squid 存储缓存对象的目录。它将存储最大 100MB 的对象，超过这个值，旧的对象将被剔除出缓存并被新的对象取代。最后两个数指定 Squid 将在主缓存目录下创建多少个子目录。在本例中，它将在主目录下创建 16 个子目录，每个子目录又包含另外 256 个子目录。

```
# cache_dir ufs /var/spool/squid 100 16 256
```

出现这种子目录方案是因为许多文件系统在访问包含大量文件的目录时非常慢。比如，如果 Squid 的每个子目录包含大约 100 个缓存的文件，那么缓存中总文件数会超过 400000 个，当其中一个文件被浏览器请求时，每次为了找到所需数据存储在何处，该主机需要在目录索引节点遍历搜索 400000 个条目。把缓存分成小块，那么在搜索给定文件时，需要检索的索引节点的数量将小得多，其性能就不会随着缓存的增加而显著降低。

因为通常磁盘空间比带宽便宜得多，所以把缓存增大一点。

```
cache_dir ufs /var/spool/squid 512 16 256
```

保存该配置文件，然后启动 Squid。在 Red Hat 上，首先应该用 chkconfig 创建启动链接。

```
$ sudo /sbin/chkconfig --level 2345 squid on
```

现在，在 Red Hat 上，可以用“sudo service squid start”来启动它；而在 Ubuntu 上则用“sudo invoke-rc.d squid restart”来重启它。在 Red Hat 上，指定的缓存目录将在 Squid 首次启动的时候创建。而在 Ubuntu 上，当安装了该软件包后，该目录便被创建了。

11.4.2 客户端配置

剩下要做的就是配置 Web 浏览器，让它使用 Web 代理。在 Firefox 中，选择菜单“编辑”➤“首选项”，接着选择“网络”标签页，然后单击“设置”按钮。选择“手动配置代理”并为我们的代理输入地址和端口号，如图 11-23 所示。

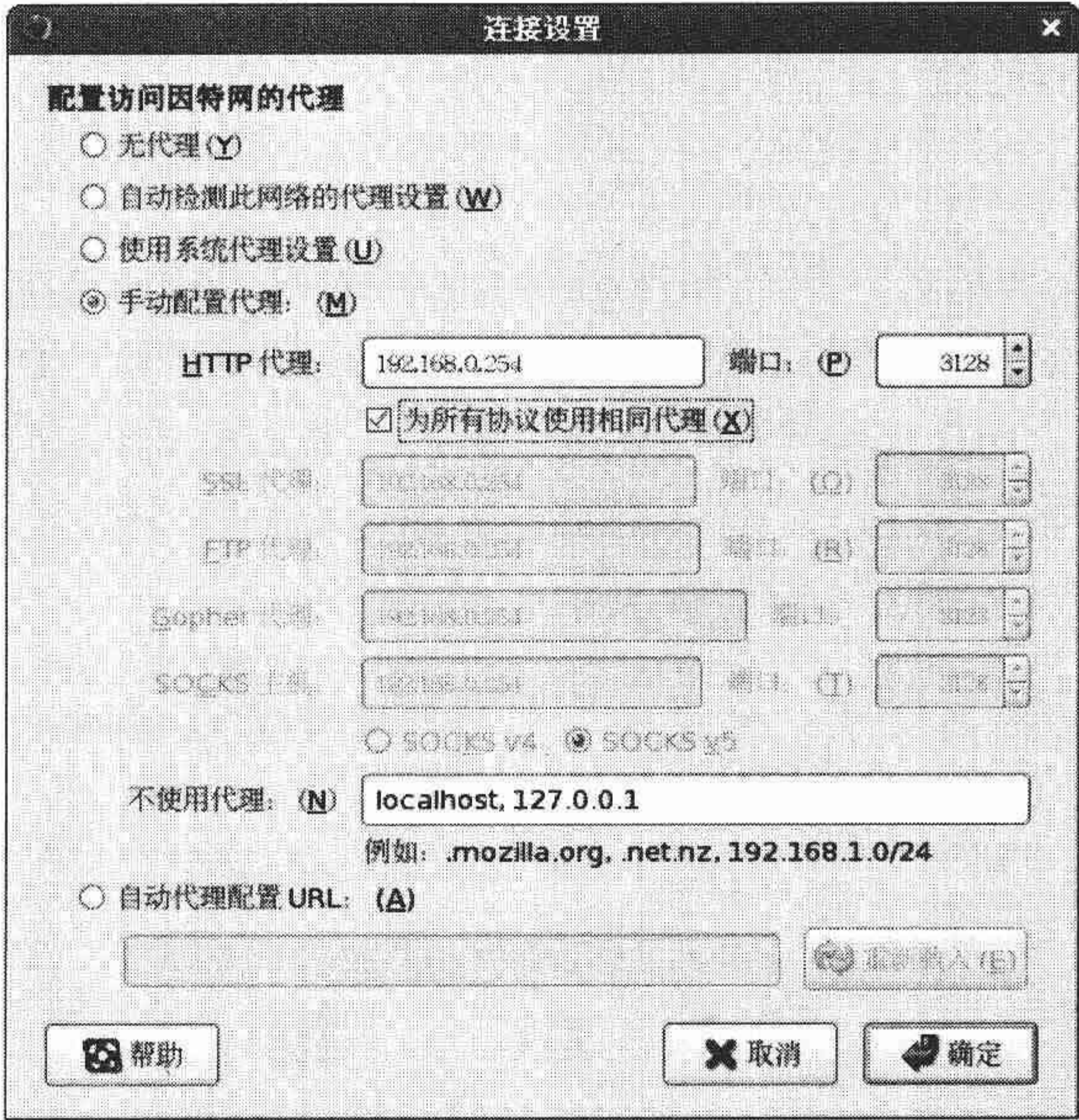


图 11-23 在 Firefox 中设置代理服务器

同样，通过在“不使用代理”列表中添加两个局域网域（192.168.0.0/24 和 192.168.1.0/24）告诉 Firefox 不要给局域网内的任何主机使用缓存。接着关闭 Firefox 网络设置对话框，并且访问网站。可以通过查看 Squid 的访问日志证实它正在被使用，这些日志存在文件/var/log/squid/access.log 中。

11.4.3 透明性

如果不想让用户为了使用 Squid 缓存而更改它们的代理设置，可以把 Squid 作为一个透明代理来运行。可以通过防火墙规则把所有向外的 Web 通信都重定向到透明代理。正在访问

Web 的浏览器并不知道它们正在使用缓存服务器，并且也不需要明确地配置任何 Web 浏览器。

为了把现在设置的代理服务器变成透明代理，必须在配置文件中做两个小的改动。需要为每个 `http_port` 指令添加 `transparent` 项。

```
http_port 192.168.0.254:3128 transparent
http_port 192.168.1.254:3128 transparent
```

重启 Squid 后，在网关主机中添加防火墙规则。这些规则应该拦截所有访问远程网站的连接，并且把它们重定向到代理。在网关把数据包发送到 Internet 之前，要更改其中的目标地址和端口号。这是网络地址转换的一种形式，可以用 Netfilter 中的 `nat` 表来实现。既然想在数据包发送之前更改它的目标地址，我们将在 `PREROUTING` 链中做相关配置。

■注：在第 6 章中涉及防火墙的内容。

为了更改数据包中的目标地址，需要使用 DNAT 目标对象。DNAT 目标对象要求 `--to-destination` 参数。`--to-destination` 参数依次要求一个目标 IP 地址和一个可选的端口号。如果没有指定端口号，它将保持不变。

```
$ sudo iptables -t nat -A PREROUTING -i eth0 -d ! 192.168.0.0/16 -p tcp --dport 80 ➡
-j DNAT --to-destination 192.168.0.254:3128
$ sudo iptables -t nat -A PREROUTING -i eth1 -d ! 192.168.0.0/16 -p tcp --dport 80 ➡
-j DNAT --to-destination 192.168.1.254:3128
```

通过 `-i` 选项可以指定通信进入所在的网络设备，这样从 Internet 到我们本网络主机的请求就不会受到影响。我们还希望只拦截 Web 请求，所以指定规则仅匹配到 80 端口号的 TCP 通信。最后，该规则应该只作用于局域网之外的主机，因为需要确保到局域网内 Web 主机的通信不受影响。通过 `-d` 选项，指定该规则只能配置目标地址不在 192.168.0.0 到 192.168.255.255 地址范围内的通信（加一个感叹号作前缀表示非的意思）。

■注：如果透明代理不和网关处于一台主机，应确保它能直接访问远程站点，而不会被重定向到它自身。

11.5 小 结

在这一章，我们展示了如何把 Linux 主机用作带有 MySQL 支持的灵活的 Web 服务器，以及如何通过安装 Web 服务来利用它。现在，读者能够做以下事情。

- 创建和管理虚拟站点。
- 为 Apache 增加功能模块。
- 基于主机名或者用户名密码来控制访问。
- 创建和管理 MySQL 数据库和用户。
- 安装和配置第三方 Web 应用程序。
- 配置 Web 代理来节约带宽费用和提高速度。

下一章将通过添加文件和打印共享功能来扩展服务器的功能。

第 12 章 文件和打印共享

文档的共享和打印是企业最常见的应用之一。本章将向读者介绍如何使用 Linux 完成文档的共享与打印。Linux 提供多种方法来共享信息，它可以通过一个叫做 Samba 的集成工具与微软 Windows 系统的客户端共享信息，也可以使用一个名为网络文件系统（Network File System, NFS）的工具在 Linux（及其他 Unix）主机之间共享文档。

不过，我们的讨论将不局限于基本的文件共享范围，还将向读者介绍 Linux 何以成为一个廉价而又高效的文档管理平台。在现代企业中，文档共享是一个重要的应用，然而，文档管理（从文档的创建到发布）是文档的安全与控制流程中极其重要的一部分，它所提供的管理粒度是简单的文件系统共享无法实现的。为此，将向读者介绍一个免费的文档管理系统（DMS）——KnowledgeTree，它可以完全替代微软的 SharePoint 应用程序。本章最后将介绍如何在 Linux 主机上配置打印和打印服务。我们讨论两个方面的问题：如何打印主机上的文档，以及怎样使主机充当打印服务器。Linux 上的打印很容易实现，Red Hat 和 Ubuntu 共用一个通用的工具集，该工具集简化了在 Linux 主机上打印的实现和对打印的管理。

本章最终的目的是提供一个完备的、现代的以及安全的共享数据网络。Samba 将作为主控制器，允许用户访问 Linux 主机的用户主目录和打印机服务。我们将介绍如何使用 NFS 服务器跨 Linux 主机挂载文件系统，以实现在 Linux 之间共享数据。公司的所有文件交由 DMS 控制，由 DMS 提供安全的 HTTP 访问、版本控制和工作流应用。本章将介绍如何使用上述功能。

12.1 使用 Samba 和 NFS 共享文件

文件共享是指在用户之间共享文档的技术。与其每个人把文档保存在各自的主机上通过 E-mail 之类的工具传递它们，不如把这些文档集中存放在一个地方并允许员工以一种可控而安全的方式访问它们。这样做是大有好处的，因为它不仅减少了在不同主机上保存同一份文档的副本的数量，还使得文档的访问控制和备份操作变得更加简单。集中式存储的应用程序必须有一个共享目录以供其他主机用户访问其内容，文档共享技术正好可用于此。

在 Windows 主机上，可以看到有一个共享目录被映射为网络驱动器。当登录到 Windows 主机时，Windows 就会通过运行登录脚本的域控制器为我们分配一个共享目录，如果需要，也可以手动创建共享目录，还可以把共享目录挂载到 Linux 主机上。Linux 系统可以方便地挂载 Samba 共享目录和 NFS 共享目录，但是，在它们的共享目录下无法运行未经修改的 Windows 可执行文件。

■注：对于某些使用 wine 开发的 Windows 应用程序，也许可以在 Linux 下成功运行。winehq 开源社区致力于集成一组 Windows 应用程序，使其在 Linux 操作系统上运行。wine 还有个 winehq 兄弟组织 CodeWeavers (<http://www.codeweavers.com>) 所提供的支持版本。

在 Linux 上可采用多种方式实现文件共享，所选方式取决于我们在访问控制、安全以及所支持的客户端等方面的要求。Linux 能为传统的文件共享提供一个良好的平台，因为它允许在客户端计算机上挂载网络驱动器。这就意味着可以通过设立一个集中式的文件存储服务器来访问共享的公共数据。实现传统的文件共享的两个主要工具是 Samba 和 NFS。Samba 用于微软的 Windows 操作系统，而 NFS 用于 Linux 或 Unix 系统。

Samba 为访问共享目录和使用打印机等服务提供身份认证。NFS 本身不提供任何身份认证，但它可以集成到支持身份认证的 Kerberos 域中。这是个复杂的安装过程，我们并不打算在本书涉及。

接下来将向读者详细介绍 Samba。

12.2 Samba

Samba 是应用于 Linux 系统的一种文件共享和打印服务。它采用标准的客户机-服务器模型，使用一个守护进程接收来自网络客户端的请求。Samba 基于公共因特网文件系统 (Common Internet File System, CIFS) 协议和服务器信息块 (Server Message Block, SMB) 协议。这些协议控制微软的 Windows 文件系统之间的通信，并支持 Samba 兼容微软 Windows 系统的桌面客户端和域名服务。

Samba 和微软 Windows 系统

大多数公司都需要一个文件服务器。现在，Samba 的主要竞争对手是常用的 Windows 服务器。Windows 服务器基本上就是一个类似于 Samba 的文件和打印服务器。微软目前并未在它的 Active Directory LDAP 服务器中提供高级的集中式用户管理系统，而是把 Active Directory LDAP 服务器紧密集成到文件和打印服务器中。如今，与 OpenLDAP 结合使用，Samba 就可以具备 Windows 服务器的大部分功能。为此，需要多花些力气把 Samba 与 OpenLDAP 良好地集成在一起，这么做是否值得取决于我们的实际情况。

Samba 即将发布 Samba 4 服务器版本，它将和微软的 Active Directory 文件与打印服务器不相上下。Samba 和 OpenLDAP 的紧密集成使得我们可以方便地安装一个单一的认证服务器，用以文件共享、Web 访问和 E-mail。

接下来介绍如何使用 Samba 建立一个简单的文件共享及打印服务器来支持在同事之间共

享文档。读者将学到怎样安装一个 Samba 域服务器，这样用户就可以通过认证并把他们的用户主目录挂载到各自的主机系统上。另外，我们还将介绍如何建立部门内共享，并把它挂载到自己的主机系统上。

首先安装软件。Red Hat 系统和 Ubuntu 系统使用相同的软件包来安装和运行 Samba。

在 Red Hat 主机上，输入以下命令。

```
$ sudo yum install samba-client samba-common samba system-config-samba
```

在 Ubuntu 主机上，相应的命令如下。

```
$ sudo aptitude install samba-client samba-common samba system-config-samba
```

samba-common 软件包可能已经安装在读者的主机上了，不过重新安装它并不会导致任何问题。

安装好这些软件包后，就可以开始 Samba 的配置。我们可以通过 GUI 界面程序或者编辑配置文件来配置 Samba。Ubuntu 和 Red Hat 都以 `/etc/samba/smb.conf` 作为 Samba 的配置文件。

在本文的配置示例中，我们把 Samba 主机搭建成一个域控制器。它将共享以下内容，即用户的主目录、临时目录 `tmp`、部门内共享目录 `sales`。

用户账户的认证信息可以存储在 Samba 主机上，接下来将会对此做介绍。Samba 也可以从 LDAP 服务器上查询用户的认证信息。默认的配置项并不指定集成 LDAP 与 Samba，因此必须手动在 `smb.conf` 文件中设置该选项。本章将向读者介绍这两种认证类型，但是直到第 16 章才会真正讨论如何安装 LDAP 服务器。

我们将把下面的配置文件用于本文安装的 Samba 服务器，并简要解释每一项设置的作用。首先，把原来的配置文件重命名为 `smb.conf.bak` (`sudo mv /etc/samba/smb.conf /etc/samba/smb.conf.bak`)。列表 12-1 显示的是我们为 Samba 服务器所创建的配置文件 (`sudo vi /etc/samba/smb.conf`)。

列表 12-1 Samba 配置文件/etc/samba/smb.conf

```
[global]
    workgroup = EXAMPLE
    server string = Samba Server Version %v
    netbios name = au-fileserver-1
#===== network info =====
    interfaces = lo eth0
    hosts allow = 127. 192.168.
#===== logging =====
    log file = /var/log/samba/%m.log
    max log size = 50
#===== domain info =====
    domain master = yes
    domain logons = yes

    local master = yes
    os level = 33
    preferred master = yes

    security = user
    passdb backend = tdbsam

    logon path = \\%L\Profiles\%u
```



```
logon script = %u.bat

#===== user administration =====
add user script = /usr/sbin/useradd "%u" -n -g users
add group script = /usr/sbin/groupadd "%g"
add machine script = /usr/sbin/useradd -n -c "Workstation (%u)"
-M -d /nohome -s /bin/false "%u"
delete user script = /usr/sbin/userdel "%u"
delete user from group script = /usr/sbin/userdel "%u" "%g"
delete group script = /usr/sbin/groupdel "%g"

#===== printing info =====
load printers = yes
cups options = raw
printcap name = cups

#===== services offered =====
[homes]
comment = Home Directories
browseable = no
writable = yes
valid users = %S
valid users = EXAMPLE\%S

[printers]
comment = All Printers
path = /var/spool/samba
browseable = no
guest ok = no
writable = no
printable = yes

[netlogon]
comment = Network Logon Service
path = /var/lib/samba/netlogon
guest ok = yes
writable = no
share modes = no

[profiles]
path = /var/lib/samba/profiles
browseable = no
guest ok = yes
read only = no

[tmp]
comment = temporary directory
path = /tmp
public = yes
writable = yes
printable = no

[sales]
comment = shared sales directory
```



```

path = /data/staff/sales
readonly = yes
public = no
browseable = yes
valid users = +sales
write list = jsmith, bsingh
force create mode = 0770
force directory mode = 0770
create mask = 0770
directory mask = 0770
force group = sales
force user = exbackup

```

该配置文件分为两部分，首先是[global]配置选项，然后是一些特殊的配置选项，如[tmp]、[home]等。[global]部分定义了一些能影响整个服务器配置的选项。而在[tmp]和[home]中的特殊配置只能影响它们的目标服务。

现在，浏览一下整个[global]部分。首先看到的是 **workgroup** 选项。如果读者是个 Windows 用户，应该很熟悉该选项。**workgroup** 选项是在 Windows 网上邻居上出现的一个工作组名称。工作组（**workgroup**）就是共享信息的计算机集合。通常，一个工作组中的计算机并没有一个集中的身份认证，而是各自负责自身的用户认证。换言之，每台主机都持有一个用户列表，列表中的用户拥有访问该主机的权限。

当工作组中的某一个主机成为主域控制器（**primary domain controller, PDC**）时，就可以进行集中的身份认证。加入到该域的主机就可以用 PDC 来认证用户的访问。如果主机不是该域的一员，它依然能共享资源，但是必须持有自己的身份认证和访问列表。

在此设置工作组时，不要把主机设置为 PDC，而只能把它设置为工作组的一员。下面设置工作组名称为 **EXAMPLE**。

```
workgroup = EXAMPLE
```

下一个配置项是 **server string**。**server string** 是交由 Samba 服务显示的一段描述性文字。其中 **%v** 是一个特殊变量，它代表 Samba 服务器的版本号。在本文的例子中，Ubuntu 主机上的 **%v** 是 3.0.28，而在 Red Hat 上的是 3.0.28a。

接下来需要配置 **NetBIOS**。**NetBIOS** 是一个本地广播协议，它用来处理主机之间的连接信息。**NetBIOS** 信息在 Windows 服务器中根据名称匹配 IP 地址，有点类似于 DNS。**NetBIOS** 协议本身用在名称服务、会话（**session**）服务以及数据报服务器中。正如列表 12-1 中所示，**netbios name** 设置为 **au-fileserver-1**，可以用该 **NetBIOS** 名称指代我们的主机。

■注：如果读者对 **NetBIOS** 感兴趣，想对它做进一步的了解，请阅读“<http://en.wikipedia.org/wiki/NetBIOS>”的内容。

下一部分的配置内容是控制网络接口和 IP 地址的访问。使用 **interfaces** 选项，可以设置 Samba 在哪个网络或者网络设备上监听服务。**lo**、**eth0**、**bond0** 和 **192.168.0.1** 这些设置都是有效的，要根据网络情况进行设置。如果主机在一个网络接口上绑定了多个网络地址，可以指定 Samba 服务只在这些地址中的其中一个地址上监听连接。

举个例子，假设 **eth0** 网络接口设置为下列两个 IP 地址和网络——**192.168.0.1/24** 和

192.168.10.1/24。如果启动 Samba 服务并指定 `interfaces = eth0`，Samba 服务将在 192.168.0.1 和 192.168.10.1 两个 IP 地址上启动，并为 192.168.0.0/24 和 192.168.10.0/24 这两个网络提供服务。我们也许并不希望 192.168.10.0/24 网络上的主机访问该 Samba 服务器，所以不要指定 `eth0` 作为 Samba 服务器的网络接口，而是更为简洁地指定 192.168.0.1 作为 Samba 服务器的 ip 地址，这告诉 Samba 只在 192.168.0.1 地址上启动服务。

在本文例子中，我们不关心哪些网络能访问该 Samba 服务器，并希望 Samba 服务在 `lo`（环回）设备和 `eth0` 上都有效。

```
interfaces = lo eth0
```

■注：可以使用 `netstat-ltup` 命令来验证 Samba 服务在 `lo`（环回）设备和 `eth0` 上是否都有效。该命令将列出和网络接口绑定的应用程序和服务。

通过使用 `hosts allow` 选项指定哪些主机能够访问 Samba 服务，可以对 Samba 做进一步的访问控制。在本文配置中，环回网络和 192.168.0.网络可以访问 Samba 服务。127.和 192.168.0. 这些记号相当于分别指定了子网掩码 127.0.0.0/8 和 192.168.0.0/24。这里也可以用整个 IP 地址指定个别的主机，还可以用 `EXCEPT` 子句排除特定的主机。举个例子，如果允许 192.168.0. 网络上的所有主机访问 Samba 服务，而除去一台地址为 192.168.0.15 的不良主机，可以使用以下的 `hosts allow` 设置。

```
hosts allow = 127. 192.168.0. EXCEPT 192.168.0.15
```

还可以用类似于 `headoffice.example.com` 和 `gateway.example.com` 的全局域名（fully qualified domain names, FQDN）指定个别主机，在这里就不做介绍了。另外 `hosts deny` 设置项列出我们禁止访问该 Samba 服务的某些主机和网络。

为了使 192.168.0.0 到 192.168.30.0 网段内的主机可以访问 Samba 服务，可以指定以下选项。

```
hosts allow = 127. 192.168.
```

接下来设置日志信息。首先，定义日志文件。日志文件存放在 `/var/log/samba` 目录下。每台主机使用变量 `%m.log` 作为自己的日志文件名。其中 `%m` 是指代计算机名的特殊变量。通过指定 `log size=50`，设置 Samba 每当日志文件增长到 50KB 的时候重新覆盖它。`desktop1` 主机的日志文件是 `/var/log/samba/desktop1.log`。

用以下的配置定义我们的域信息。正如之前所提到过的，一个域可以为域中的主机提供集中式身份认证。设置域需要一台 PDC 主机或中央主机，并且同一个域中只能有一台中央主机。有 Windows 背景知识的读者对此将会很熟悉。

设置以下配置信息。

```
domain master = yes
domain logons = yes
local master = yes
os level = 33
preferred master = yes
security = user
passdb backend = tdbsam
```


“domain master = yes”意味着 Samba 服务器具备 PDC 的作用。设置 “domain logons = yes”表明要求 Samba 服务器处理来自域中主机的认证请求。

nmbd 服务器提供 IP 名称服务之上的 NetBIOS 服务。为此，它必须成为本地广播网络组的 Master Browser。设置 “local master = yes”说明我们的主机将成为 Master Browser（当服务启动时，如果竞争取胜，则成为 Master Browser）。“os level = 33”代表主机在竞争当中成为本地 Master Browser 的机会大小，或者说是期望值。“preferred master = yes”这行意味着在主机启动 nmbd 服务的时候，它将强制举行竞选并给自己赋予成为 Master Browser 的最大期望值。

■注：Master Browser 是一台主机，负责收集来自其他网络段中的 Browser 的服务器通告。这些 Browser 为客户端提供共享资源列表。请阅读 “<http://support.microsoft.com/kb/188001>”上的内容，获取关于 Master Browser 的进一步讨论。

现在，在默认的 Samba 服务器中，“security = user”用来设置连接到 Samba 服务器的任何用户需要在该主机上拥有的合法的账户和密码。其他选项能让我们加入同一个域中或同一个 AD 域中的其他 Samba 服务器，读者可以查看手册页获得更详细的内容。“passdb backend = tdbsam”表明用本地的 Samba 密码数据库做用户认证。这时其他用于认证服务的合法设置项是 smbpasswd 和 ldapsam。tdbsam 指向本地主机上的一个数据库文件，叫做/etc/samba/passdb.tdb。该文件属于 root，出于安全上的考虑，它的访问权限为 600（只有 root 用户可以读写它）。

接下来，“logon path = \\%L\Profiles\%u”指明客户端的移动配置文件存放在哪里。%L 就是服务器上 NetBIOS 的名称，而 %u 是用户进行认证时使用的用户名。如果不需要移动配置文件，logon path 等同于一个空字符串（比如 “logon path = ”）。

欲使 logon path 生效，用户必须对其有读写访问权限。在服务器主机上，profile 目录通常位于/var/lib/samba/profiles，而不是在用户的主目录下。[profile]共享服务定义了更多的配置项，包含了该服务的更多细节。可以把独立的启动脚本放在 netlogon 目录（位于/var/lib/samba 目录）下。在该配置文件中，已经指定了 logon script 指令，而[netlogon]服务也定义了更多的指令。

Samba 变量

是不是难以忍受这些 %L 参数？这里介绍 Samba 的一些标准变量的替换写法，以下是在手册页中找到的变量替代写法。

%U: 会话名称（客户端自定义的用户名，不必与客户端的用户名相同）。

%G: %U 所属的主要组的名称。

%S: 当前服务的名称，如果该服务有名称。

%h: 运行 Samba 的因特网主机名。

%L: 服务器的 NetBIOS 名称。该变量允许我们根据客户端的请求来更改服务器的设置。服务器可以有双重身份。

%M: 客户端主机的网络名称。

%D: 当前用户所在的域名或工作组名。

%H: %U 所指用户的主目录。

这里并没有介绍所有的变量，要想查看全部变量，请使用 man 命令打开 smb.conf 页面。

我们在接下来的配置中，指定如何管理主机上的用户。当在 Samba 服务器上添加、更改

或删除用户、用户组和主机时，可以通过执行一些脚本来完成。当把一台主机加入域中时，Samba 使用命令 `add machine script` 为主机添加一个计算机账户。类似地，如果往域中添加用户，Samba 将使用命令 `add user script`，如此等等。

```
add user script = /usr/sbin/useradd "%u" -n -g users
add machine script = /usr/sbin/useradd -n -c "Workstation (%u)" -M -d /nohome -s /bin/false "%u"
```

■注: Red Hat 主机所接受的添加用户的命令和 Ubuntu 所接受的添加用户命令有细微的区别。如果使用 Ubuntu 主机，在之前列出的脚本“`add machine script`”这一行中，必须删除 `-M` 选项。在 Red Hat 主机上，`useradd` 命令的 `-M` 选项将促使该命令不为新用户创建主目录；而在 Ubuntu 主机上，该特殊选项是无效的，主目录在默认情况下不会被创建。

如果需要，我们可以确保在用户被创建时，可将他们加入一个不同于 `users` 的特定组。举例而言，通过把 `add user script` 这行中的 `-g users` 改为其他的用户组，比如 `-g staff`，就可以把新建的用户加入到 `staff` 用户组。同样地，如果需要用 LDAP 管理用户列表，就要执行完全不同的脚本了。这些脚本由 `smbldap-tools` 软件包提供，如下所示。

```
add user script = /usr/sbin/smbldap-useradd -m "%u"
```

在这个实例中，`smbldap-useradd script` 脚本负责向认证服务添加用户，这里的 `%u` 代表用户名。

Samba 和 LDAP

如果读者有一台现成的 LDAP 服务器或者打算以后架设一台，并打算把 Samba 服务器用来做 PDC，然后把现有的 Samba 用户移到 LDAP 认证系统，那么这将花费读者不少的力气。如果读者只想把 Samba 作为文件共享服务器，则可以简单地通过给 LDAP 服务器指定 `passdb backend`，从而使 Samba 利用 LDAP 做身份认证。

```
passdb backend = ldap://ldap.example.com
ldap admin dn = cn=samba,ou=meta,dc=example,dc=com
ldap suffix = dc=example,dc=com
ldap user suffix = ou=People
ldap group suffix = ou=Group
```

如果已经有一台 LDAP 服务器，则需要设置 `ldap admin dn` 的详细配置，使其具有合适的识别名 (DN) 和恰当的 LDAP 前缀等。我们将在第 16 章介绍 OpenLDAP。

下一步设置打印信息。该设置用于把 `[printers]` 共享服务连接起来。“`printcap name = cups`”这行告诉 Samba 使用 CUPS 作为打印服务。“`load printers`”指令告诉 Samba 加载 CUPS 默认配置的打印机。“`cups options`”指令用来给 CUPS 服务器发送选项信息，其默认值是 `raw`。我们可以在 CUPS 的手册页面找到其他的选项说明。

现在，可以查看列表 12-1 中 Samba 配置的实际服务了。Samba 把一些服务看作“特殊”的服务，如 `[global]`、`[homes]` 和 `[printers]`。在配置文件中，这些服务按段区分，以中括号 `[]` 开始。Samba 对待特殊服务和 `[sales]` 与 `[tmp]` 那些普通服务是有所不同的。

正如我们所设想的，当使用 `[printer]` 服务接收一个文件时，Samba 会对它做出不同的处理，不同于那些通过 `[tmp]` 服务所接收的文件。Samba 具有某些只和“特殊”服务中的设置相关的

功能。比如，定义在[printer]中的 path 指定了一个假脱机目录，当我们的打印机用[printer]服务把输出数据定向到假脱机目录时，Samba 就会自动把它传递给 CUPS 打印，而在另一个服务中为 path 设置该值却不会触发这一行为。

如果要设置自定义的服务，则需使用特殊服务和用户自定义服务中的常见指令集。表 12-1 中列出了例子中用到的指令。读者可以用这些指令来修改或创建自己的服务。

表 12-1 Samba 服务指令

指令	描述
path	定义所描述的共享资源的路径（比如/tmp）
browseable	描述共享资源是否在浏览器列表中可见
comment	为共享资源指定一段描述性文字
writable	指出该共享资源可写，与只读相反
readonly	表示该共享资源是只读的
printable	允许创建假脱机文件并把它放到共享区。只使用于打印服务
guest ok	指出不需要密码就可以访问某个共享资源。默认情况下是 no（不允许），也叫做 public
valid users	指定一个可以使用某个服务的用户列表
write list	指定一个可以“读/写”共享资源的“用户/组”列表，而不管 readonly 设置
force user	为访问某个共享资源的所有连接分配一个默认用户
force group	为访问主机的所有连接分配一个默认的组
force create mode	对创建的文件强制赋予 Unix 权限
force directory mode	对创建的目录强制赋予 Unix 权限
create mask	对创建的文件设置 Unix 权限
directory mask	对创建的目录设置 Unix 权限

可以使用这些服务指令设置路径以及设置共享资源的访问权限。读者可以在 man smb.conf 手册页面找到更多可用指令的详细解释。

正如之前所提到的，[home]是一个特殊的共享服务。Samba 查看用户认证的详细信息，从而找出用户根目录的路径（或者在需要的时候创建一个目录）。该服务的 valid users 需要设置一个已连接并通过认证的用户，该用户由客户端的服务名称所指定（客户端匹配于被请求的服务名称）。根目录被设置为可写（writable）而不是可浏览（browseable）。可浏览意味着目录会出现在服务器的 Browser 列表中，该列表类似于我们在网上邻居的浏览窗口中看到的内容。

[printers]是 Samba 提供的另一个特殊的服务。在这里指定假脱机目录“path = /var/spool/samba”。该目录不可浏览，不可写，并且不能被未经授权的用户访问（guest ok = no）。但是，printable（可打印）意味着客户端可以向/var/spool/samba 目录写入和提交用于打印的假脱机文件。

[netlogon]共享服务在 Samba 中并没有被归类为特殊服务，而在许多默认的 smb.conf 配置文件中，它却被当作一个标准服务。[netlogon]服务默认包含在 Ubuntu 和 Red Hat 的 Samba

软件包中，它为用户提供登录脚本（前面的指令“`logon script = %u.bat`”指定了[netlogon]的这一作用）。由于安全方面的原因，`shared mode` 被设置为 `no`，并且该目录是不可写的。`shared mode` 默认设置为 `yes`，该设置允许在打开文件的时候开启共享模式。另外，`guest ok` 设置为 `yes`，允许用户不通过密码认证就可以访问登录脚本。

[profiles]共享目录服务类似于[netlogon]服务，它是用户保存配置文件的地方。配置文件包括 Application Data、My Documents 以及其他 Windows 用户目录。这些目录被保存在“`logon path = \\%L\profiles\%u`”指令所指定的/var/lib/samba/profiles/user 目录下。

■注：如果在 My Documents、Desktop 或者 Application Data 目录下存有大量数据，则保存远程配置文件会使远程登录的过程变得非常缓慢。网站 http://wiki.samba.org/index.php/Samba_&_Windows_Profiles 上有一些关于该问题的提示性信息。

此外必须牢记，如果读者想要完全避免远程配置文件，则可以设置“`logon path = ''`”，这样配置文件将被保存在本地主机。

接下来介绍被广泛使用的共享目录/tmp。默认情况下并不设置该目录，但管理员通常会创建它。/tmp 共享目录允许在用户之间进行一般的文件交换。/tmp 是一个内容时常变化的目录，它时常会被清空，所以应该警告用户不要用该目录存放重要的文档。该目录有着非常宽松的权限设置，通常所有人都可以访问它。/tmp 自身有一个“粘着位”（sticky bit），这意味着用户只能删除/tmp 目录中属于自己的文件（请阅读第 3 章关于目录权限的讨论）。另外，最好不要把那些不希望被任何人看到的敏感文档放到该共享目录下。设置“`public = yes`”语句使得来宾用户可以访问该共享目录，并且所有用户都可以写该目录。

最后，建立一个名叫[sales]的共享目录。该共享目录被设置成只有 sales 下的员工才能访问。我们准备把员工共享的文档放到/data/staff 目录下，而且 sales 目录也位于该目录下。默认情况下，该目录被指定为只读，这可以阻止不相关用户的写操作。此外该目录拒绝 public 用户或来宾用户的访问。[sales]共享目录是可浏览的，所以它将出现在 Samba 服务器的共享 Browser 列表中。在此，指定一个有效的用户列表，该用户列表被设置为 sales 用户组。这里的+号代表 Samba 要遍历本地 Unix 用户/组列表。sales 组中的用户对[sales]共享目录拥有读权限。接着我们明确地指定 jsmith 和 bsingh 拥有读/写权限。同样也可以指定一个用户组的读/写权限，如果创建了用户组 sales_admins，则可以如此设置“`write list = @sales_admins`”。

最后一部分的配置确保用户创建的文件和目录具有正确的属主、用户组和所有权。我们不愿意用户把自己所拥有的文档散布在系统的各个地方，当他们想要同其他用户分享这些文档的时候，不得不再去更改这些文档的访问权和所有权。所以要求这些文件和目录的属主永远是 exbackup，同时让备份脚本也能访问这些共享目录。而且，我们打算根据部门组来设置这些共享目录的群组拥有者。这要求文件对特定的用户和组是可读和可写的，并且把目录的全部访问权限授予特定的用户和组，而不是普通的公共用户。

```
force create mode = 0770
force directory mode = 0770
create mask = 0770
```



```
directory mask = 0770
force group = sales
force user = exbackup
```

正如第 3 章所介绍的，这里设置所有文件和目录的权限是可读、可写和可执行/可访问的。该权限可能过于宽松，但这样的设置可以让我们了解如何使用权限。也可以让所创建的文件只具有 0660 权限（对所有人可读），但这会带来管理上的开销。

现在已经对 Samba 做了一个基本的配置。如果读者需要，可以添加自己的共享目录到该配置文件中。现在还需要对主机做一些设置，以便为用户服务做好准备。

首先，创建目录/data/staff，并更改它的权限。同时保证合适的用户组可以访问我们所设置的共享目录。接着，如果用户组 sales 和 staff 不存在则创建他们。

```
$ sudo mkdir -p /data/staff/sales
$ sudo groupadd -g 501 staff
$ sudo groupadd -g 502 sales
```

注：这些组 ID 都是随意的，在读者的主机上他们可能已经被使用了。读者也可以为自己的网络选择不同范围的 ID。Red Hat 上，低于 500 的 ID 值已预留给了系统账户；而 Ubuntu 上低于 1000 的 ID 值预留给了系统账户。

接下来更改目录/data/staff 和/data/staff/sales 的权限。用户组 staff 需要有/data/staff 目录的访问权限，而用户组 sales 需要有/data/staff/sales 目录的访问权限。

```
$ sudo chgrp staff /data && sudo chgrp staff /data/staff
$ sudo chgrp sales /data/staff/sales
```

现在，已经为这些目录设置了权限。还要禁止普通用户访问这些目录，并只允许特定的用户和组访问他们。

```
$ sudo chmod 750 /data && sudo chmod 750 /data/staff
&& sudo chmod 770 -R /data/staff/sales
```

上面的例子允许 staff 用户组中的所有用户访问/data/staff 目录，同时让 sales 用户组中的所有用户对/data/staff/sales 目录有读、写和删除文件的权限。sales 用户组中的用户也将在 staff 用户组中。

如果这些特定的 Samba 共享目录还没有创建，则需要创建他们，并给他们分配正确的权限。

```
$ sudo mkdir -p /var/lib/samba/netlogon && sudo mkdir -p /var/lib/samba/profiles
$ sudo chgrp staff /var/lib/samba/{netlogon, profiles}
&& sudo chmod 770 /var/lib/samba/{netlogon, profiles}
```

这里，创建了 2 个目录，然后授权 staff 用户组中的用户访问和读取这些目录的权限。

12.2.1 给 Samba 添加用户

我们已经用如上所述的方法建立了 Samba 服务器，接下来需要管理 Samba 主机上的用户账号。Samba 提供 smbpasswd 工具来管理 Samba 主机上的用户账户。该工具的命令语法如下

所示。

```
$ sudo smbpasswd [options] username
```

感兴趣的选项主要是 **-a** 和 **-d**。**-a** 用来添加用户，而 **-d** 用来禁用用户。按如下方法添加一个用户。

```
$ sudo smbpasswd -a jsmith
New SMB password:
Retype new SMB password:
Added user jsmith.
```

上面命令添加了一个新用户 **jsmith**。**jsmith** 用户需要有一个 Linux 用户账户，否则就不能添加成功。下面看看试图添加用户 **bsingh** 时会发生什么（**bsingh** 在系统中没有账号）。

```
$ sudo smbpasswd -a bsingh
New SMB password:
Retype new SMB password:
Failed to modify password entry for user bsingh
```

在添加 **bsingh** 到 Samba 服务器之前，需要把该用户添加到 Linux 主机。**root** 是我们必须添加的一个用户。当我们在 PDC 模式下使用 Samba 时，Samba 需要 **root** 用户来添加其他用户和计算机账户。

```
$ sudo smbpasswd -a
New SMB password:
Retype new SMB password:
```

在此，必须添加 **root** 账户。往域中添加主机时，也需要 **root** 账户。如果员工 **tbear** 已经离开了公司，他的 Samba 账户需要被禁用，所以删除 **tbear** 的详细信息，如下所示。

```
$ sudo smbpasswd -d tbear
Disabled user tbear
```

还要分别从 Linux 用户账户和所有 **tbear** 所属的用户组中删除他。

在完成上述操作之后，接着介绍如何向域服务器添加 Windows 主机。

12.2.2 向域中添加主机

首先，需要启动 Samba 服务器并确保配置已被加载。在 Red Hat 主机上，使用 **service** 命令；在 Ubuntu 主机上，使用 **invoke-rc.d** 命令。

```
$ sudo /sbin/service smb restart
$ sudo /usr/sbin/invoke-rc.d samba restart
```

如果启动发生什么错误，可以检查相应的日志文件。日志文件位于 **/var/log/samba** 目录中。Samba 服务器的守护进程会向 **/var/log/samba/smbd.log** 中写入日志，而客户端则向 **/var/log/samba/client-name.log** 中写入日志。根据运行的主机是 Red Hat 主机还是 Ubuntu 主机，还需要分别检查 **/var/log/messages** 文件或者 **/var/log/syslog** 文件中的任何错误信息。

现在，用本地管理员账户登录到 Windows。然后，右键单击“我的电脑”图标并选择“属性”菜单。选择“计算机名”标签页，然后单击“网络 ID”按钮。现在，已经启动了“网络标识向导”，如图 12-1 所示。

选择“下一步”继续。接着需要描述计算机的用途，如图 12-2 所示。

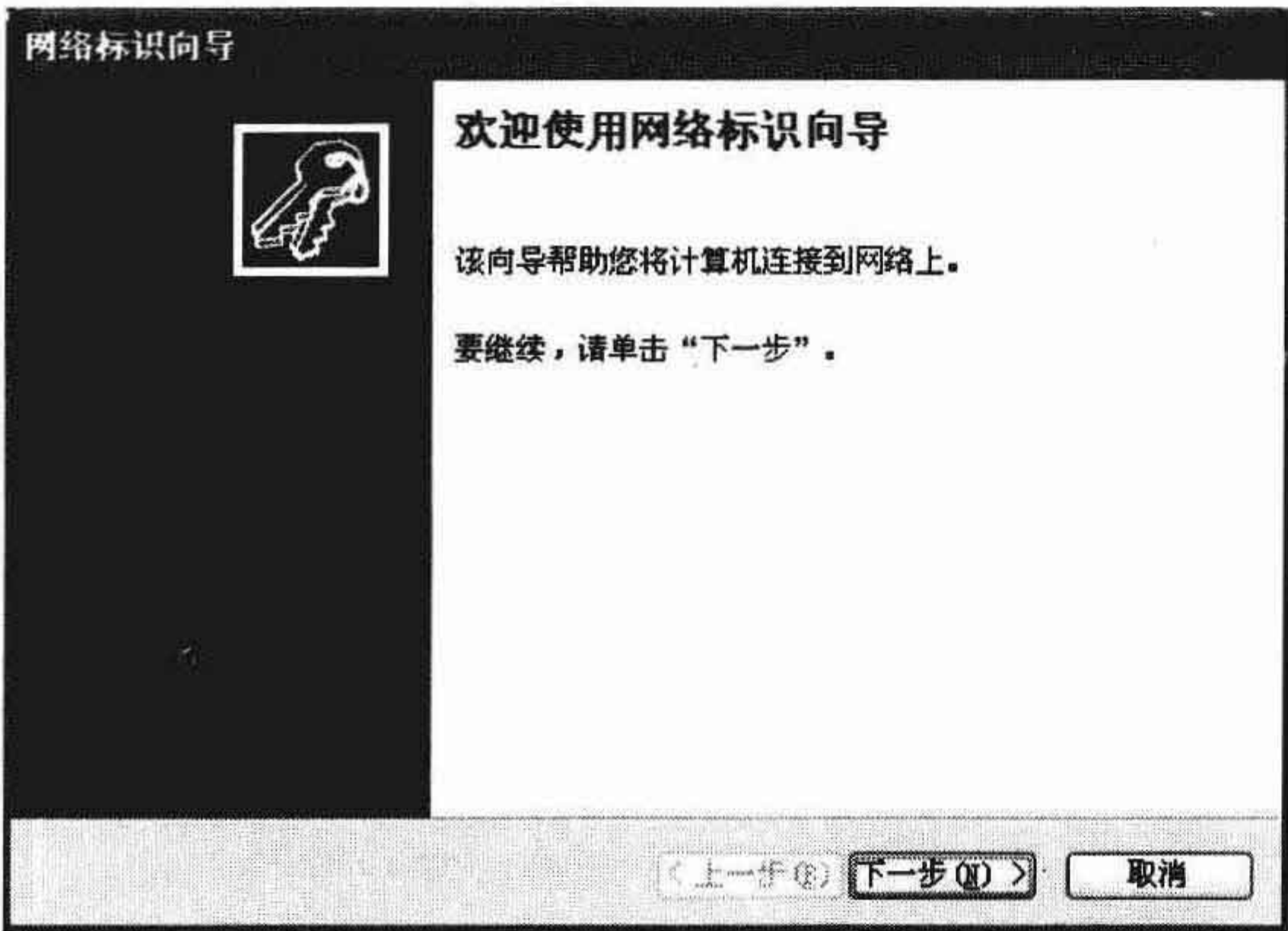


图 12-1 网络标识向导

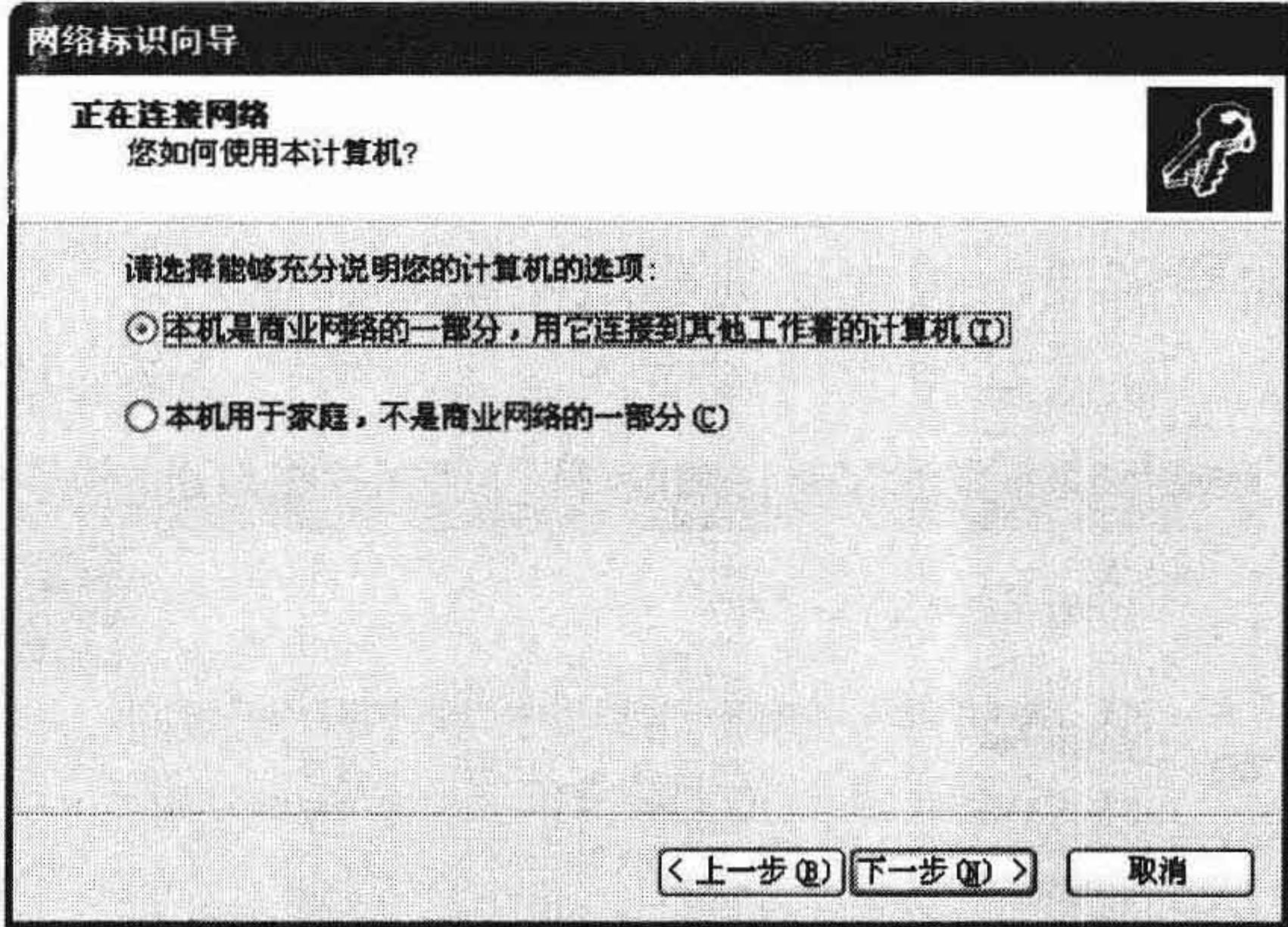


图 12-2 选择生活中计算机的主要用途

该计算机是企业网络的一部分，所以选择相应的选项并且单击“下一步”按钮继续。需要选择公司是否使用带有域的网络；在本例中，我们的企业将要使用带有域的网络——虽然企业还没有带有域的网络——所以选择使用带有域的网络。如果选择该选项，接下来需要填写已有的信息，以加入域网络。

下一步，添加用户账户信息。在本例中，使用用户 jsmith 的账户信息。该向导要求输入用户名、密码以及希望加入的域，如图 12-3 所示。

现在已经为用户 jsmith 添加了详细信息，并且打算把该用户加入 EXAMPLE 域。把用户 jsmith 加入 EXAMPLE 域是因为 smb.conf 文件的 workgroup 定义中指定了该域。

在下一个向导页，需要输入计算机名和域名并且再次单击“下一步”。使用计算机名 DESKTOP1 和域名 EXAMPLE，如图 12-4 所示。

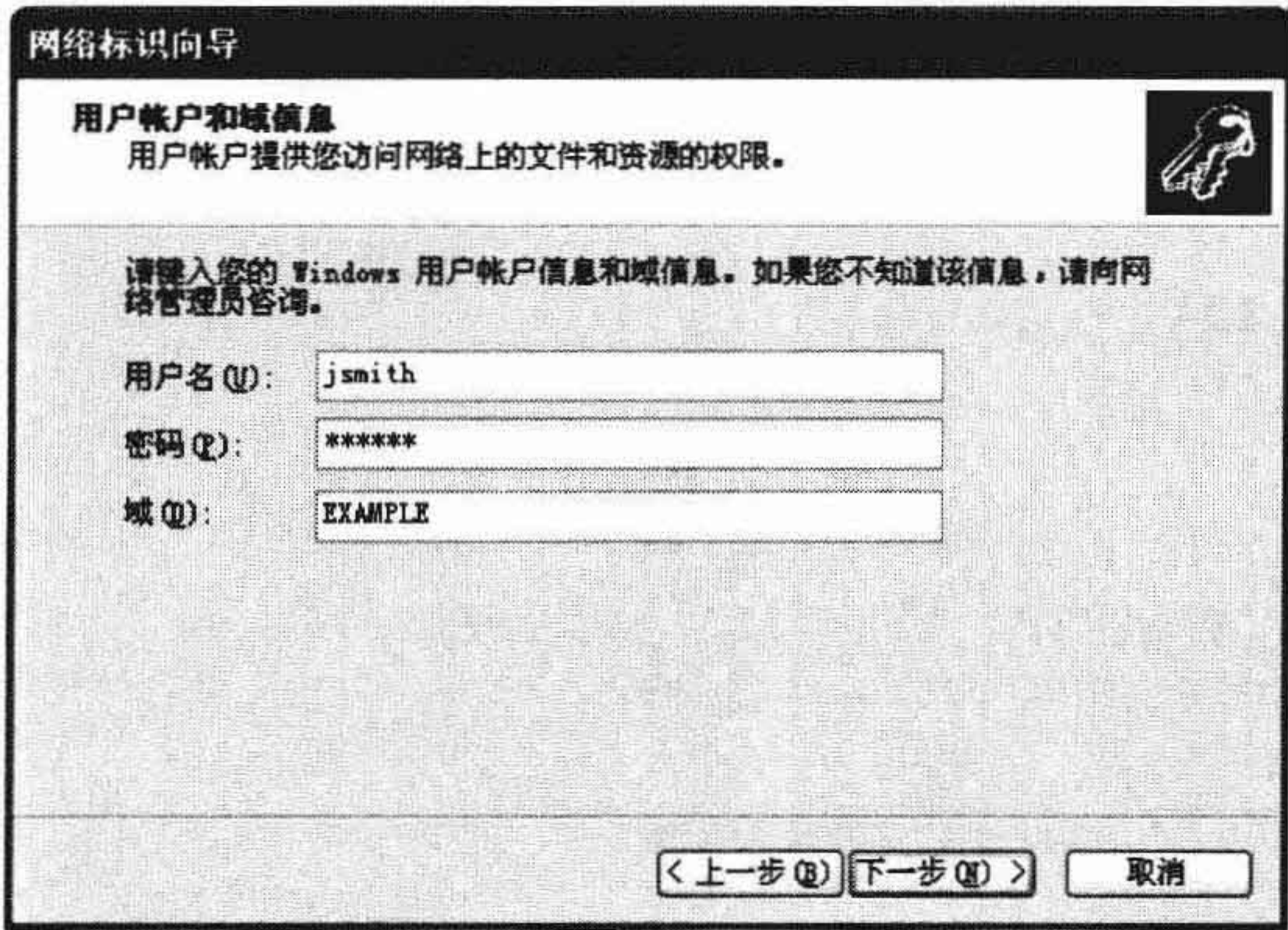


图 12-3 用户账户信息

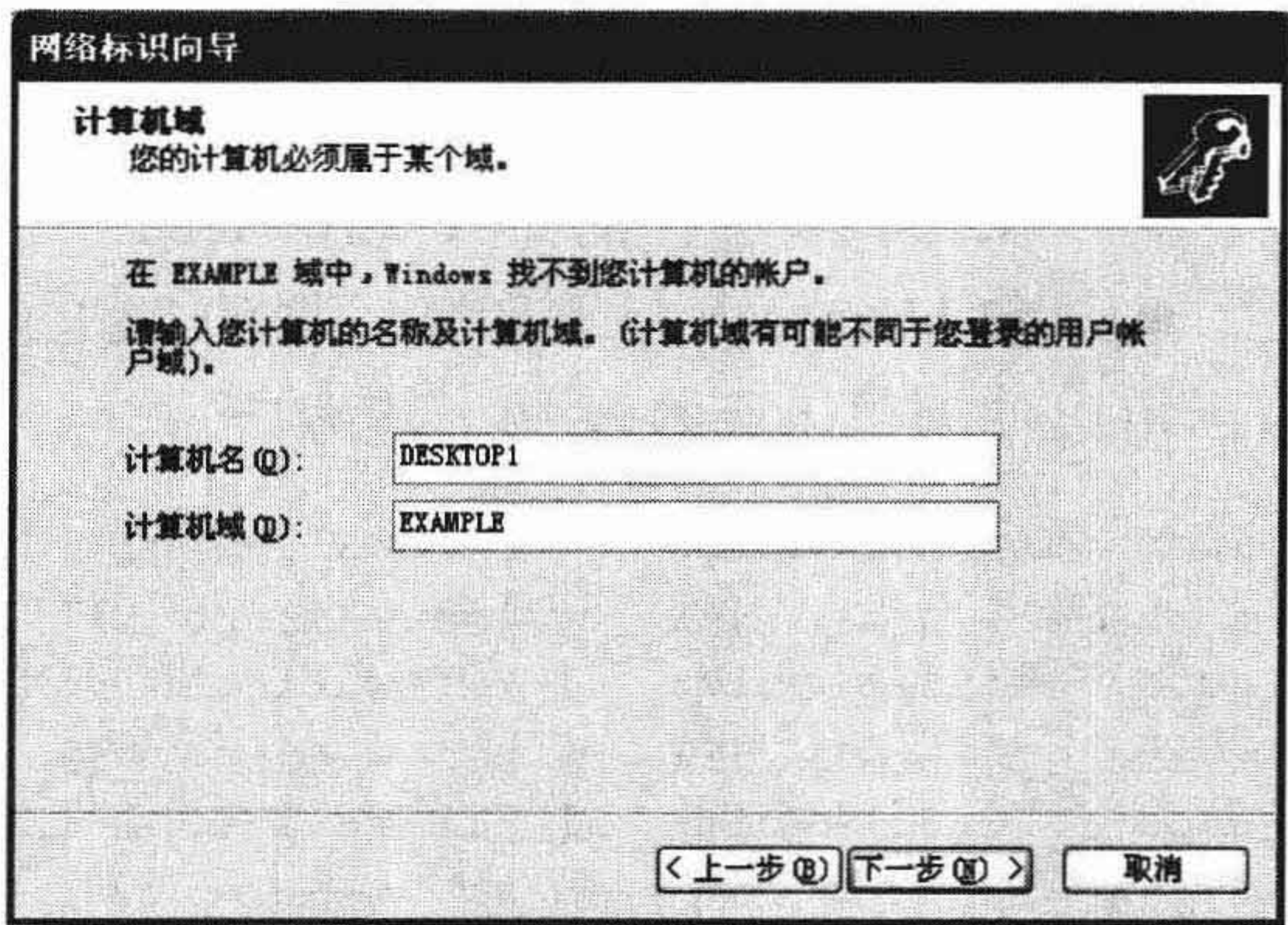


图 12-4 添加计算机名和域名

接着需要输入用户详细信息，该用户有权向域中添加主机。在本例中，该用户必须是 root

用户，密码是先前输入的 Samba 密码（见图 12-5）。

现在，由于将要向域中添加用户，可能会看到另一个窗口，如图 12-6 所示。该窗口询问我们是否希望把 jsmith 用户添加到正要加入域网络的本地计算机中，因为本地计算机还没有该用户的账户。选择不添加，因此用户 jsmith 仅能使用域网络来登录该主机，并且没有本地账号。读者的环境可能与该范例不一样，如果需要，也可以向本地主机添加用户。

在成功完成向域中添加主机时，会看到如图 12-7 所示的确认界面。

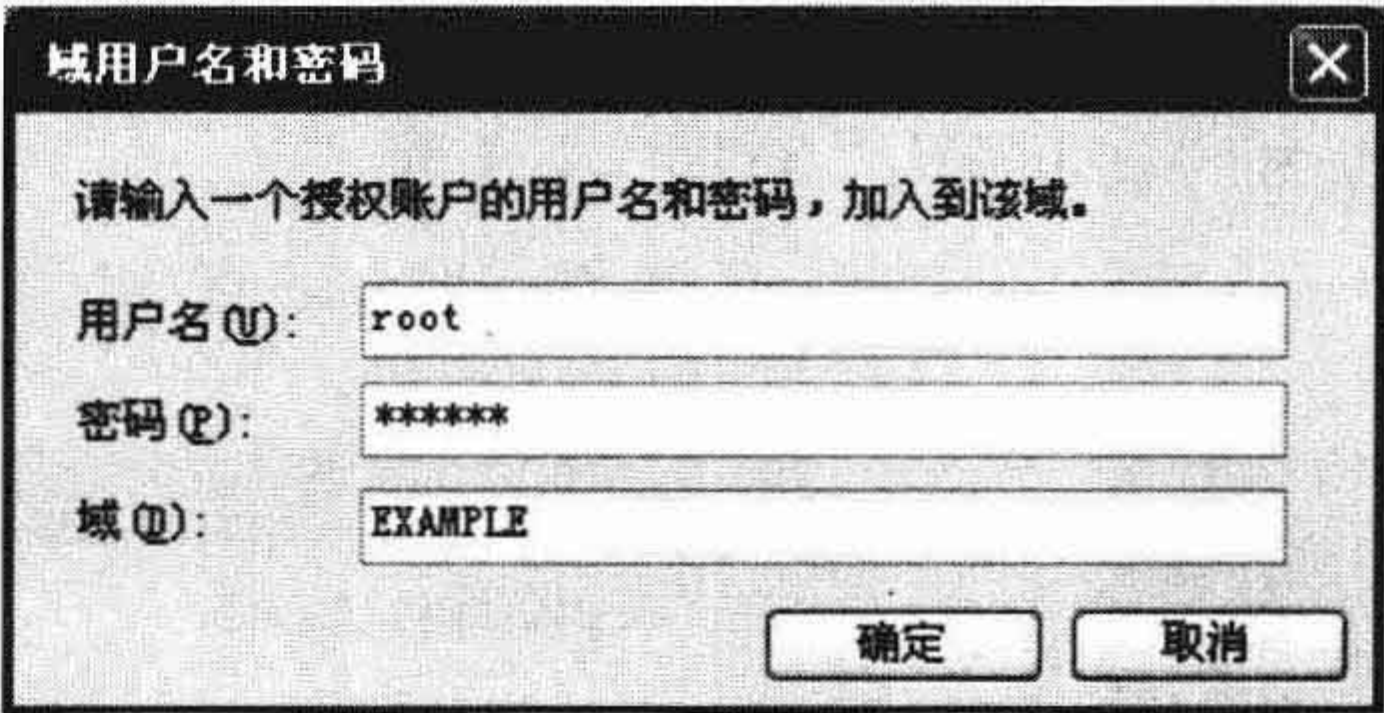


图 12-5 使用 root 用户作为管理员

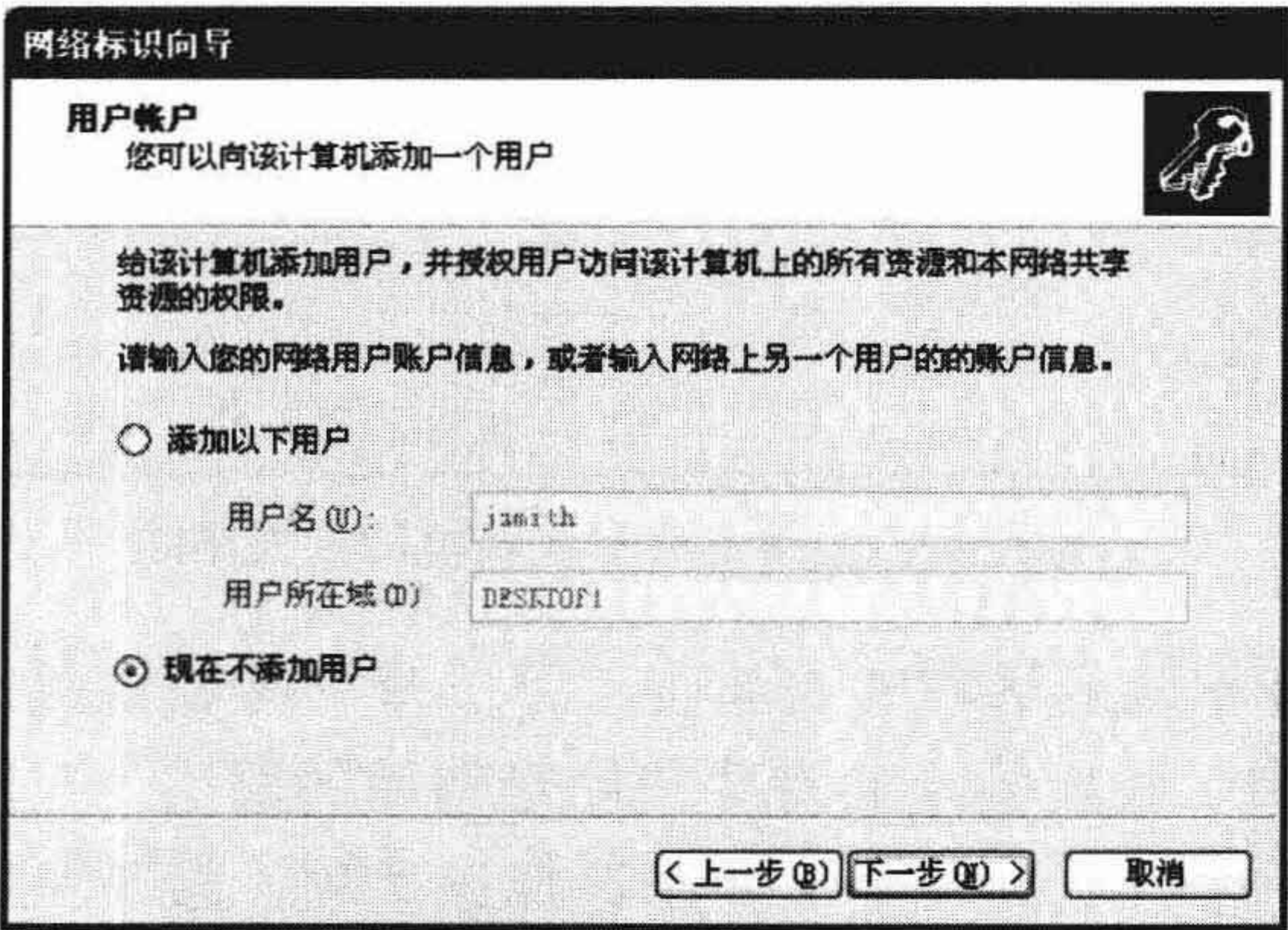


图 12-6 添加用户到本地主机的选项

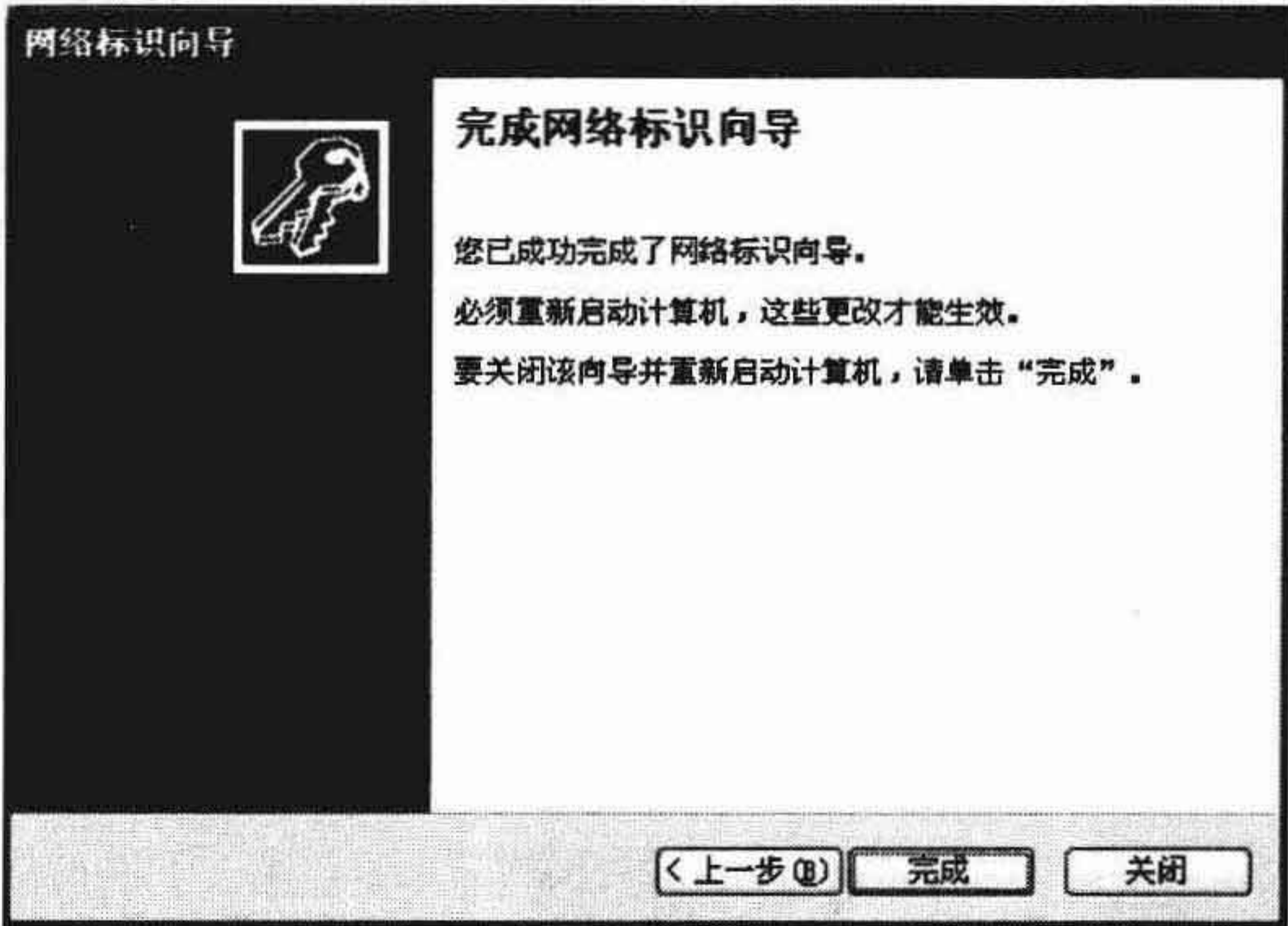


图 12-7 网络标识向导完成

现在已经把主机添加到域中。在这些设置生效之前，需要注销并且重启 Windows 客户端。接下来可以像使用任何 Windows 文件服务器一样使用 Samba 文件共享服务器了。我们将看到一个域登录界面，并可以在 Samba 服务器上存储远程配置文件。

12.2.3 Samba 所需的 IP 表规则

Samba 要求防火墙开启以下端口。

- NetBIOS 名称服务的 UDP 端口 137 和 138。
- NetBIOS 会话的 TCP 端口 139。
- 和 Microsoft 域名服务一样的 TCP 端口 445。

可以使用以下命令添加上述端口。

```
$ sudo /sbin/iptables -I Firewall-eth1-INPUT rule number -p udp ➡
-m state --state NEW -m udp --dport 137:138 -j ACCEPT
$ sudo /sbin/iptables -I Firewall-eth1-INPUT rule number -p tcp ➡
-m state --state NEW -m tcp --dport 139 -j ACCEPT
$ sudo /sbin/iptables -I Firewall-eth1-INPUT rule number -p tcp ➡
-m state -state -m tcp NEW --dport 445 -j ACCEPT
```

这里所指定的 INPUT 链表必须是 IP 表规则中所定义的，并且 rule number 优先于任何 REJECT 和 DROP 规则。

■注：第 6 章中讨论 IP 表。

12.2.4 在 Linux 上挂载 Samba 的共享目录

Linux 主机还可以用 mount 命令和 smbfs 软件挂载 Windows 的共享目录。然而，一些 Linux 发行版本并没有读写 NTFS 共享目录的能力。这是因为 Microsoft 认为这样做是一种违规的做法。Ubuntu 允许挂载 NTFS 和 FAT 文件系统，可以在 smbfs 包中找到相应的工具。要想在 Linux 主机上挂载 Samba 共享目录，需要做以下操作。

```
sudo smbmount //au-fileserver1/data/staff /data/remote/➡
-o ip=192.168.0.1,user=jsmith,dom=EXAMPLE
```

上述操作将在/data/remote 目录下挂载远程的 Samba 共享目录/data/staff。smbmount 命令指定了远程主机的 IP 地址、用户名以及工作组名称或域名。命令执行过程中需要密码，执行完毕后该共享目录就会被挂载到/data/remote 目录下。要想获知更多关于挂载 Samba 共享目录的内容，请阅读 man mount.cifs 手册页面。

12.2.5 使用 system-config-samba 图形用户界面

为了能够方便而快速地在 Linux 主机上共享文件，我们可以使用 system-config-samba 图形界面工具。该工具可以用在 Red Hat 主机和 Ubuntu 主机上，它虽然不能用于创建复杂的 Samba 配置，但是可以方便快速地创建共享目录。

通过“System”>“Administration”>“Samba”启动该工具，如图 12-8 所示。

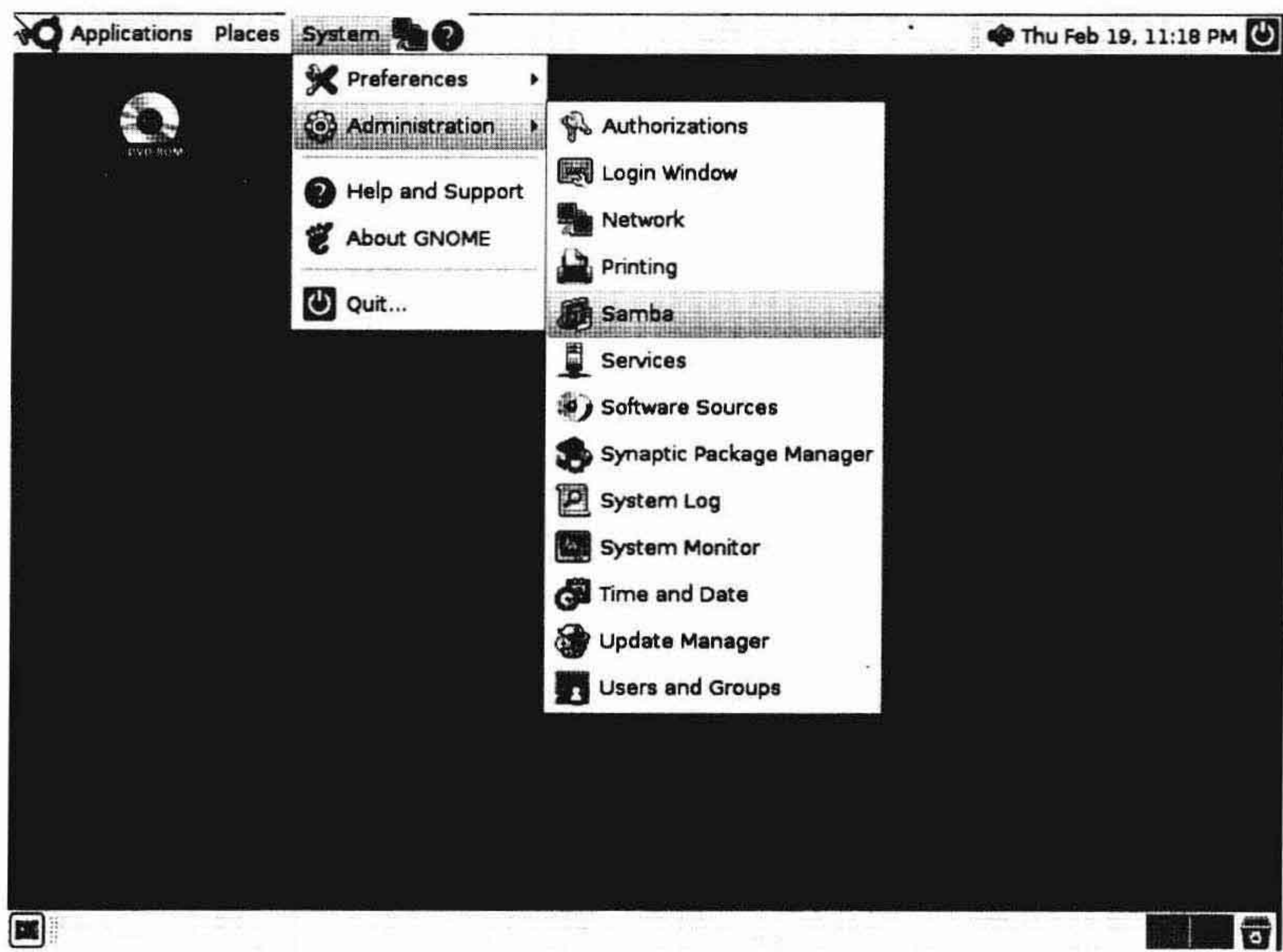


图 12-8 启动 Samba 配置工具

■注：虽然我们使用的是 Ubuntu 桌面系统来显示该图形工具，但它在 Red Hat 上也是可用的，而且界面显示和 Ubuntu 上的完全相同。

如图 12-9 所示，一旦打开该工具，打印机共享便自动生效。而且，它是可见的（比如：可浏览），其共享名是 print\$。

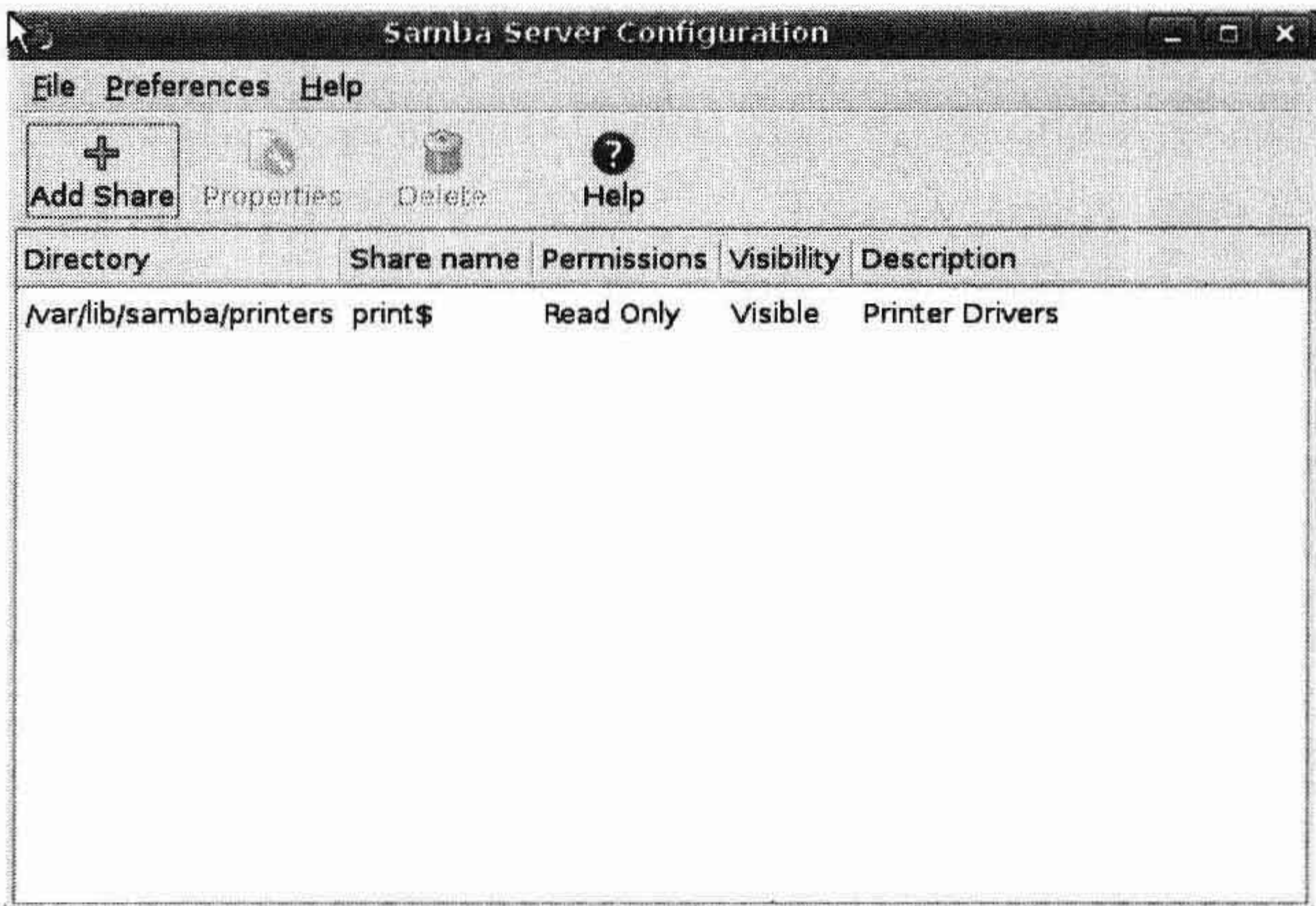


图 12-9 system-config-samba 图形用户界面

可以通过选择菜单（见图 12-10）“Preferences” ➤ “Server Settings” 更改 Samba 服务器的设置。

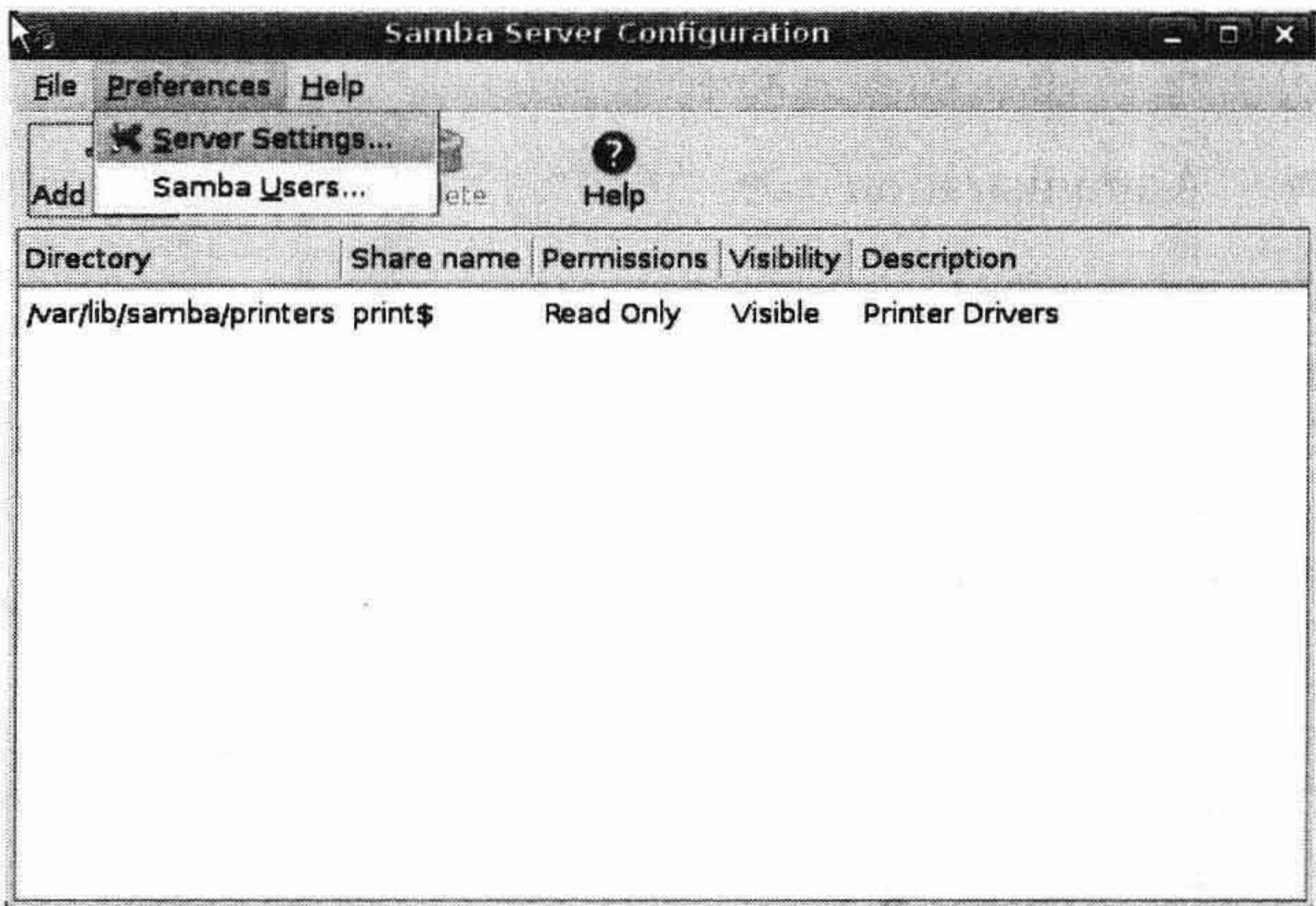


图 12-10 选择服务器设置菜单项

在该界面中，可以设置工作组名称和相应的描述。这里也可以使用一些 Samba 变量，比如表示主机名的%h，如图 12-11 所示。

如果选择了“Security”标签页，就可以选择服务器的用户认证模式。有效的 Samba 用户认证模式有活动目录服务（Active Directory Service, ADS）、域（Domain）、服务器（Server）、共享（Share）和用户（User），如图 12-12 所示。在本例子中，选择用户（User）模式。

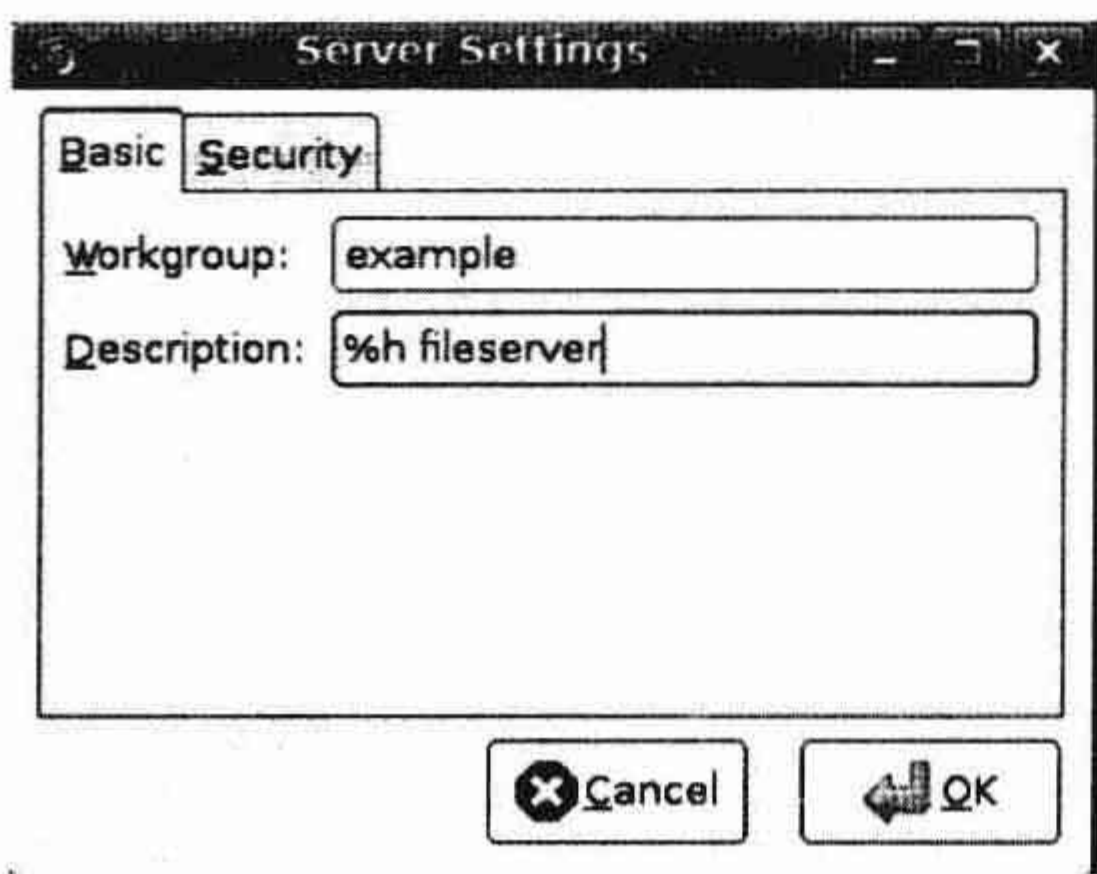


图 12-11 基本服务器设置



图 12-12 安全设置

如果有认证服务器或 Kerberos 域，也可以填写其详细信息。在图 12-12 中，选择对密码加密，同时拒绝来宾账户访问。

通过选择菜单 “Preferences” ➤ “Samba Users”，就可以添加用户账户。在图 12-13 中，可以看到已经有两个账户被添加到主机上了。这两个账户分别是 nobody 和 jsmith。

用户账户必须要在 Linux 主机上存在，才可以被添加到 Samba。我们刚刚在 Linux 主机上添加了 ataylor，现在希望把它添加到 Samba 服务器。单击 “Add User” 按钮，然后在包含 Unix 用户名的用户下拉列表框中选择为它新建的账户。

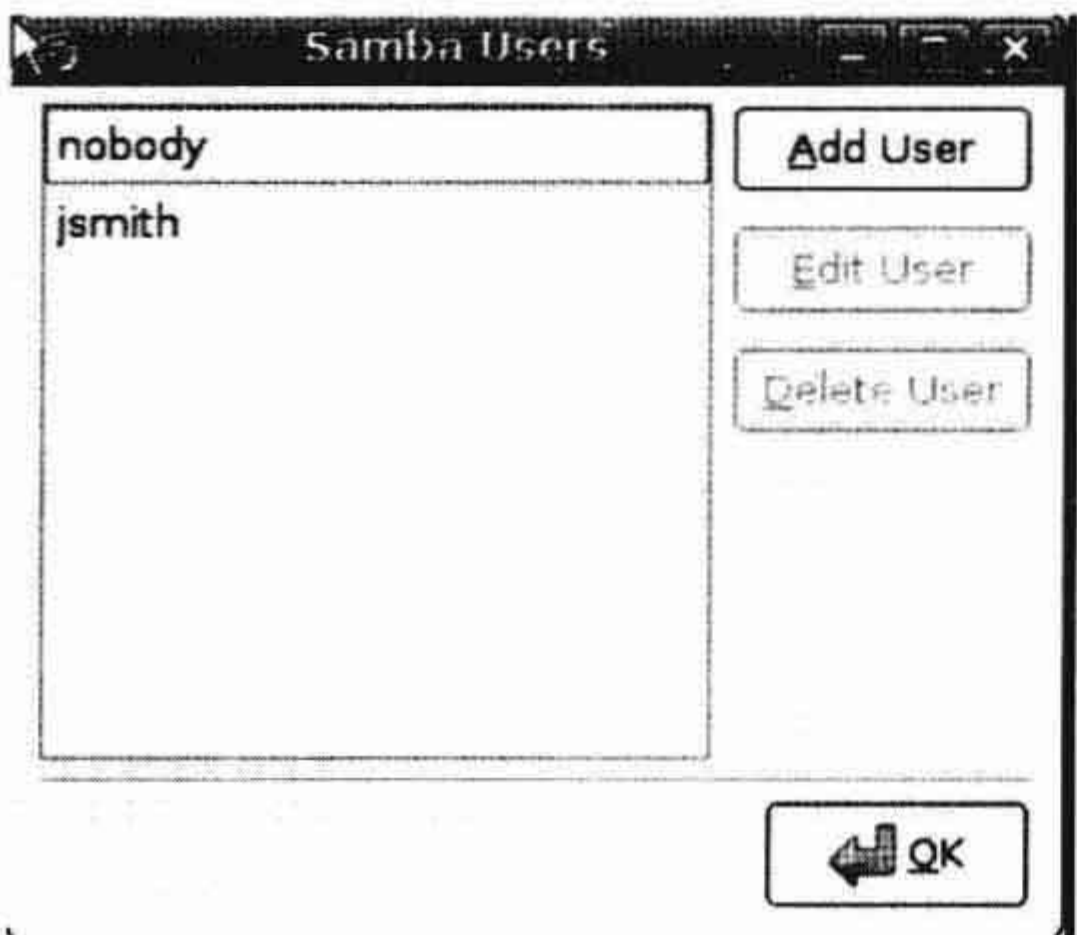


图 12-13 Samba 用户账户

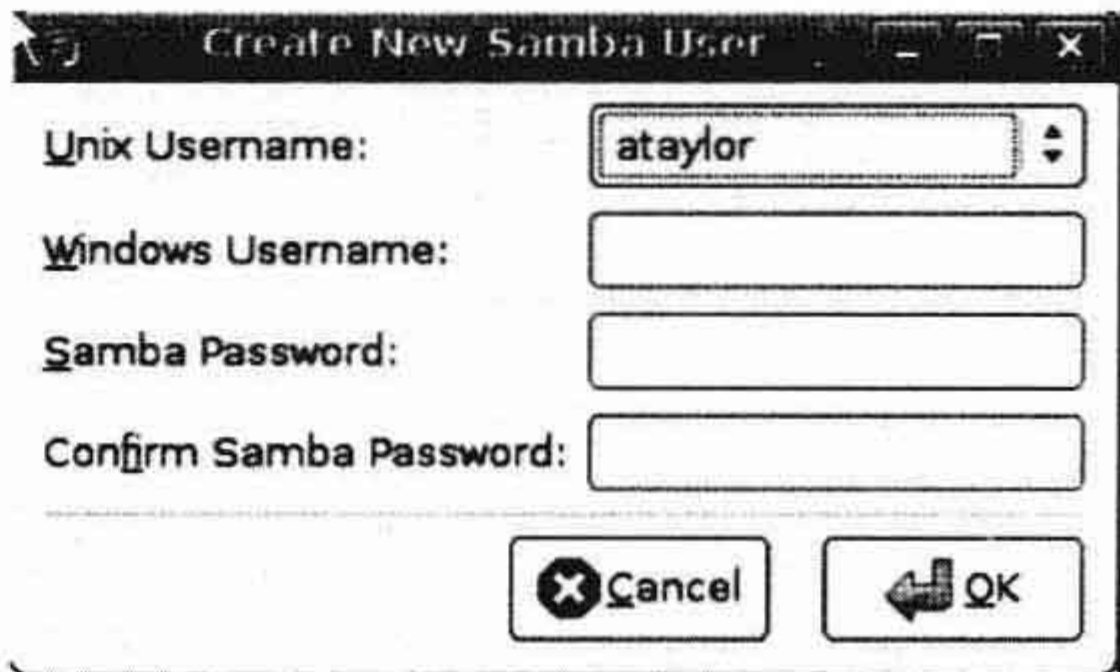


图 12-14 添加用户 ataylor

下一步，读者可能已经想到了，填写 Windows 用户名和 Samba 密码字段。图 12-15 显示了为用户 ataylor 所添加的详细信息。

单击 “OK” 按钮，往 Samba 用户数据库添加该用户。

下一步，转到创建 Samba 共享。在主窗口中，单击 “Add Share” 按钮，看到带有共享详情的 “Basic” 标签页，如图 12-16 所示。

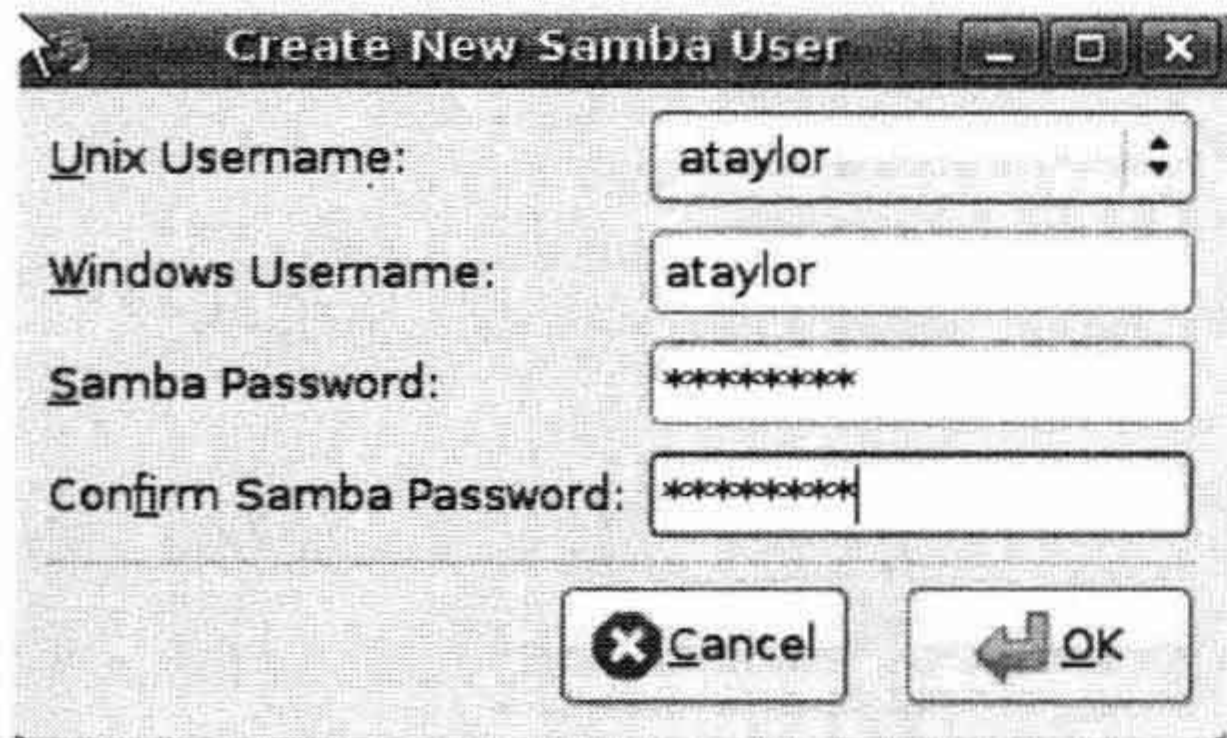


图 12-15 ataylor 用户的详细信息

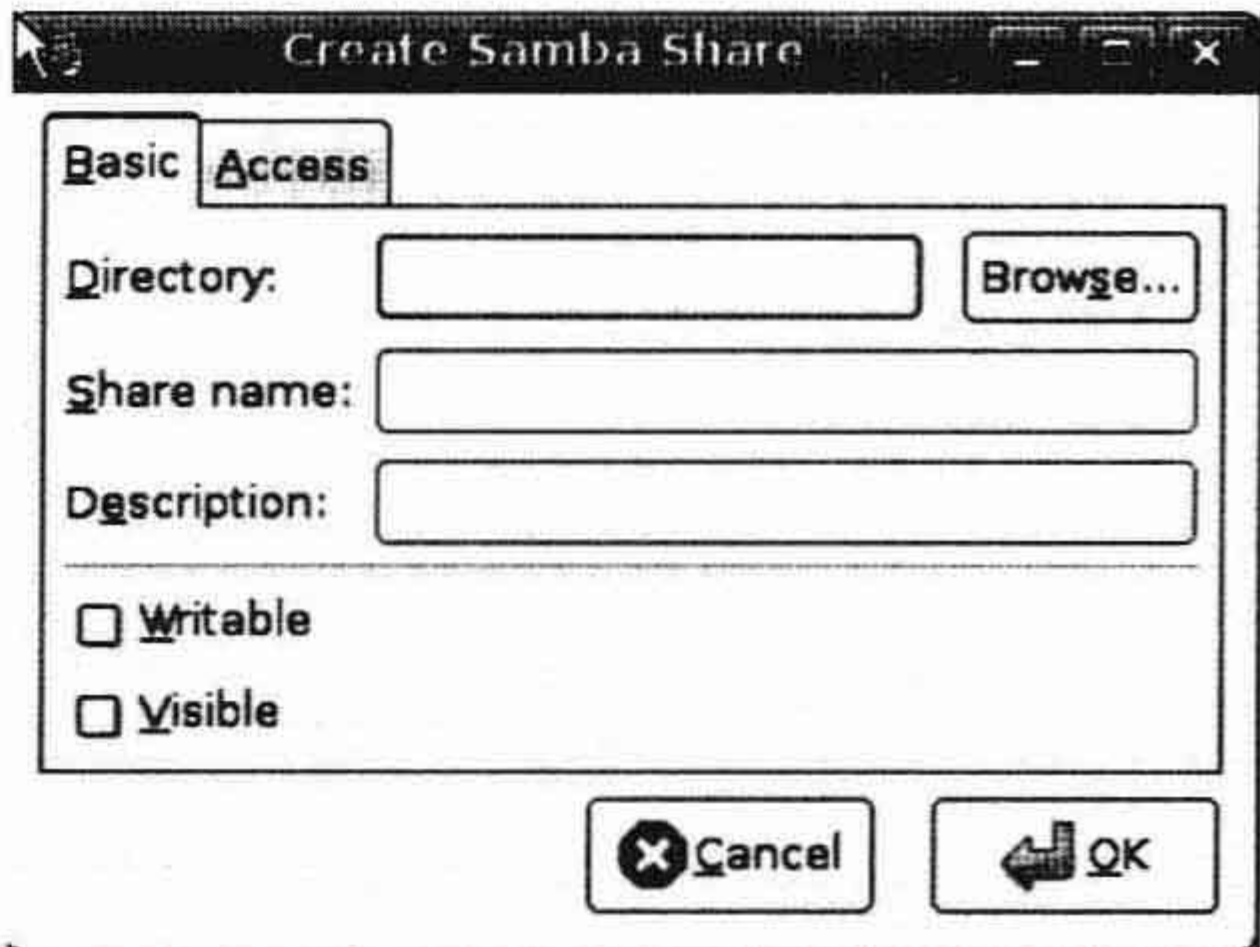


图 12-16 添加基本的共享信息

选择“Browse”按钮添加目录，弹出一个熟悉的目录浏览对话框。该对话框允许选择想要共享的目录。选择共享/data/staff/sales 目录，如图 12-17 所示。

可以看到，在图 12-18 中我们选择/data/staff/sales 作为共享目录，其共享名为 sales。把它设置为可写的（writable）和可见的（visible）（visible 是 public 或者 guest ok 的另一种表达方法）。

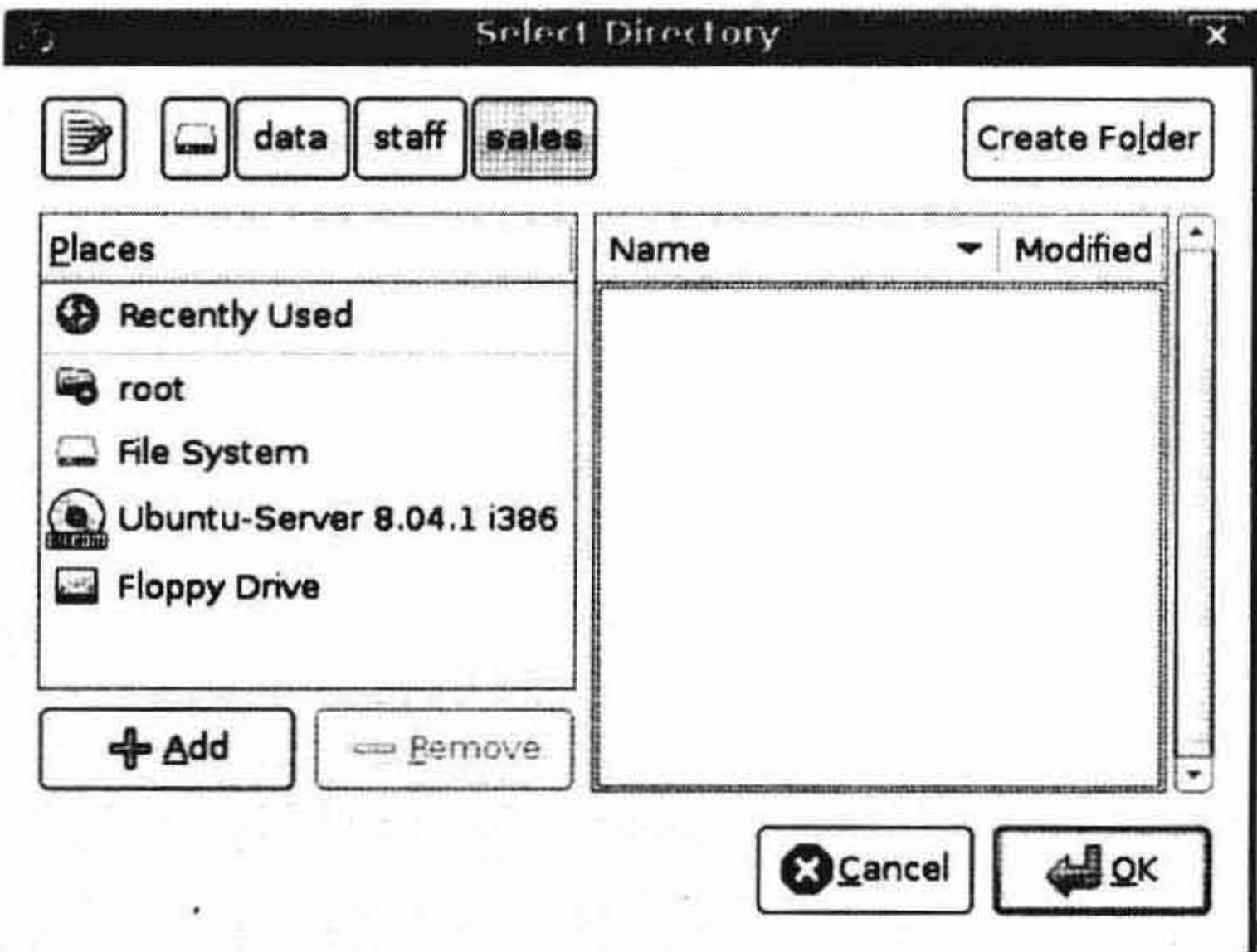


图 12-17 浏览目录

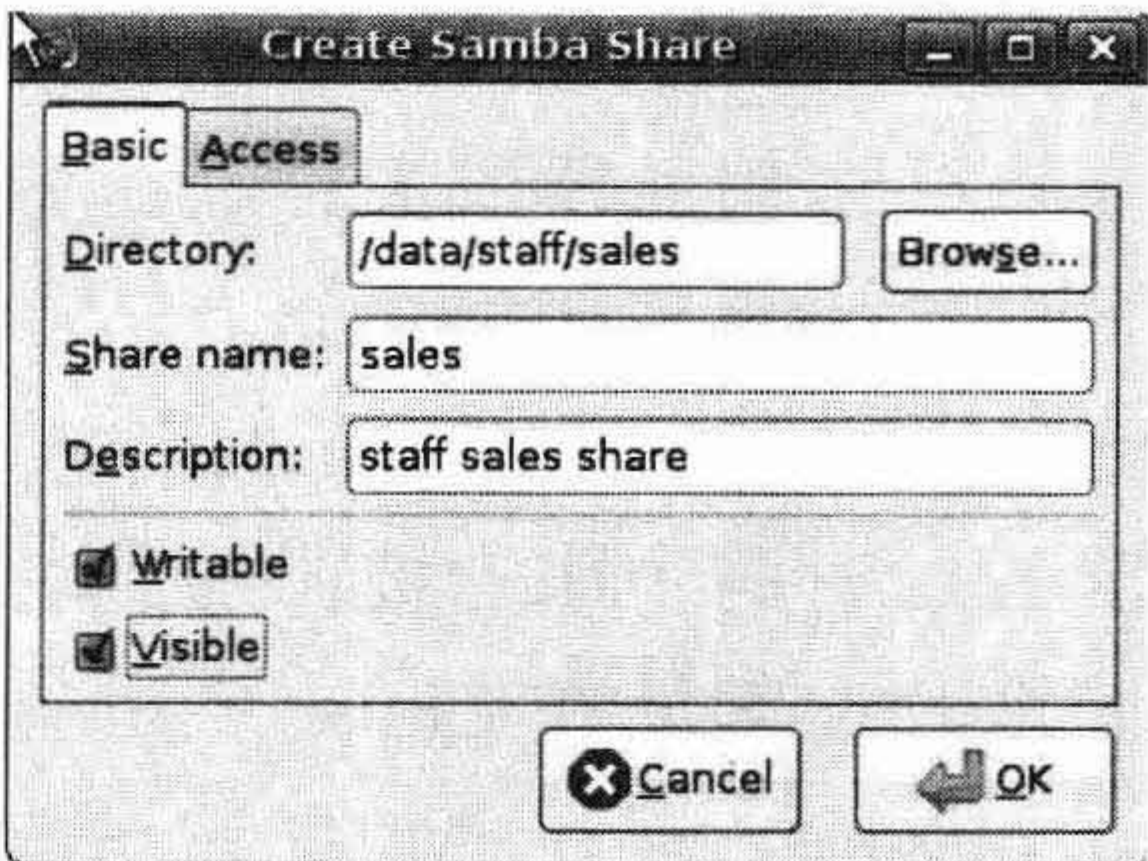


图 12-18 添加共享目录 sales 的详情

下一步，单击“Access”标签页来设置访问权限。可以看到，在图 12-19 中，单击“Only allow access to specific users”单选按钮就能添加允许访问的特定用户，也能通过单击“Allow Access to everyone”单选按钮允许所有人访问。

最后，在图 12-20 中，能看到这些设置的结果，共享目录 sales 能被用户 ataylor 和 jsmith 的桌面客户端访问。

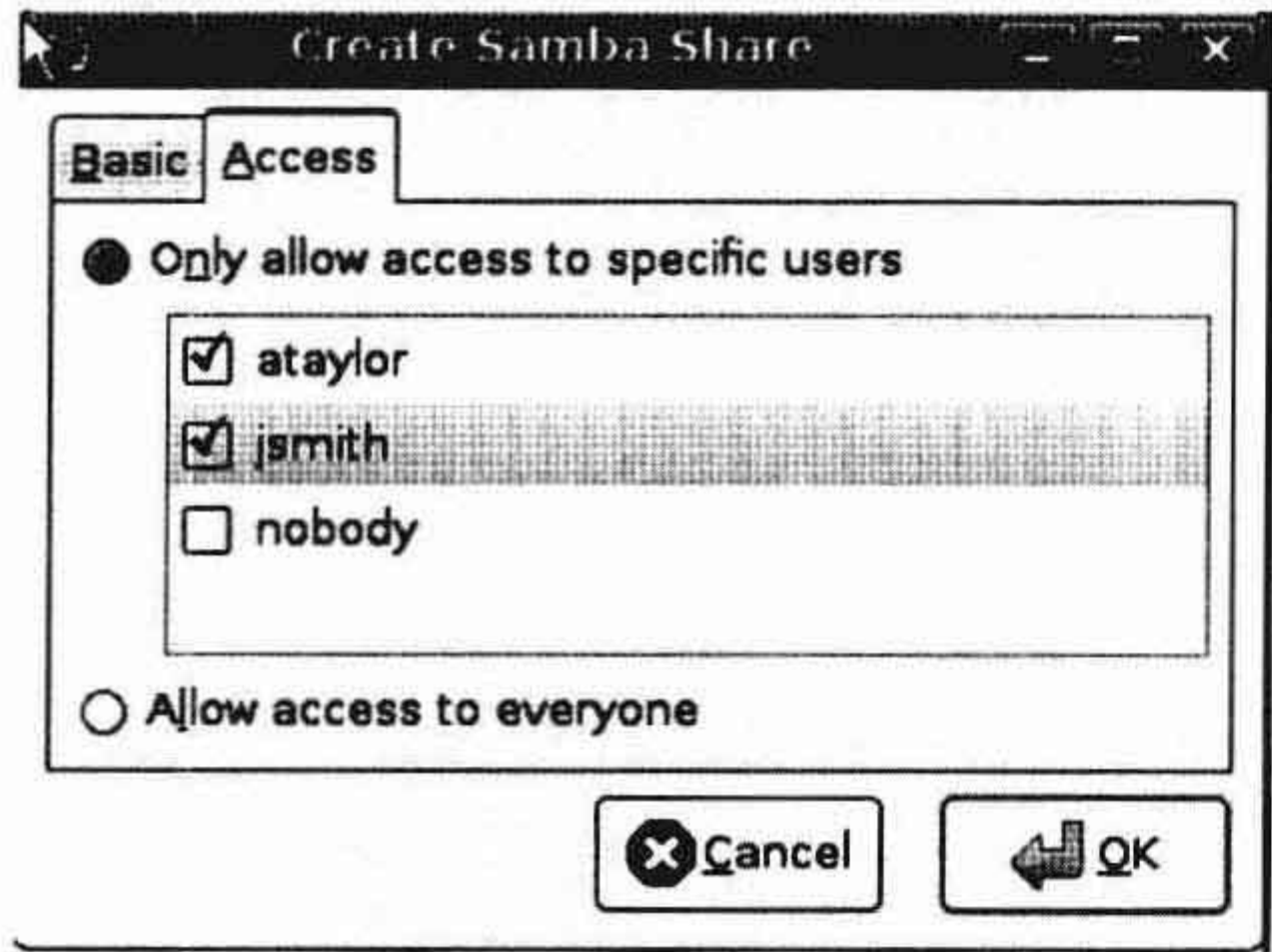


图 12-19 设置共享目录的访问权限

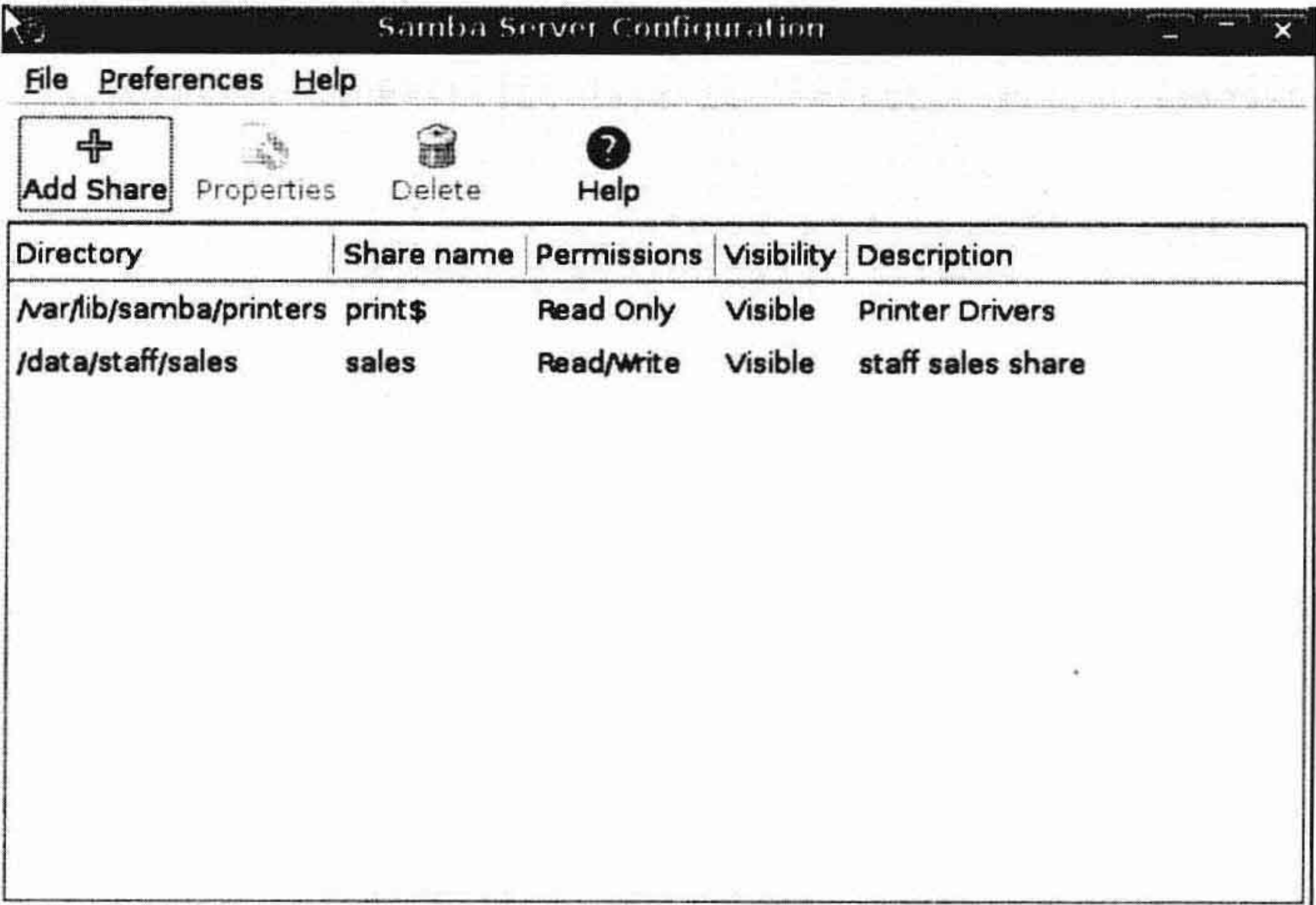


图 12-20 完成 sales 共享目录的配置

之前提到过，在给桌面客户端分发用户目录时，文件共享是个很好的办法，而 Samba 则是完成这一目的的优秀工具。还可以通过 Samba 分发普通目录来共享应用程序文件，比如数据库文件，以及用 Samba 的共享目录来保存文档，比如 word 文档、电子表格、其他文档。

■注：作为共享目录的替代方案，我们还将介绍一个可以用来管理企业文档的文档管理系统。文档管理系统提供了细粒度的、安全的版本控制，那样就不需要接触共享目录或者挂载目录。后面将介绍一个 DMS，它是“文档管理”方面的一个解决方案。

12.2.6 资源

请查看以下资源，获取有关创建 Samba 共享目录的更多信息。

- <http://www.samba.org/samba/docs/man/Samba-HOWTO-Collection/>
- <http://www.samba.org/samba/docs/man/Samba-Guide/>

12.3 NFS 共享文件：Linux 到 Linux

Linux 主机也能相互之间挂载目录，其方式和 Samba 大致一样。传统上，这是通过 Linux 之间的网络文件系统（Network File System，NFS）实现的。NFS 的最新版本（NFSv4）比之前的 NFS 版本多了许多优点。现在 NFSv4 只需要一个端口，而在这之前则需要多个端口，所以对于之前的版本而言，用户不知道它将使用哪个端口号，这使得它很难穿越防火墙。于是，由于防火墙的阻挡，许多安全方面的管理员不大愿意使用它。NFSv4 已从中进化了，它现在已经成为一个不错的网络软件。

接下来快速地展示如何在 Linux 主机之间共享文件系统——通常也叫网络挂载。在 Ubuntu 主机上，需要安装 `nfs-kernel-server` 软件包；而在 Red Hat 主机上，需要安装 `nfs-utils` 软件包。

NFS 要求防火墙开放端口号 2049。可以按如下操作添加该端口。

```
$ sudo iptables -I Firewall-eth1-INPUT -p tcp ➡
-m state --state NEW -m tcp --dport 2049 -j ACCEPT
```

读者已经在第 6 章阅读了关于 IP 表的知识。这里，也可以用其他的链表名称插入该规则，也可以把它插入到规则列表中的特殊位置。当插入完成后，还需要编辑 `/etc/exports` 文件。

NFS 从 `/etc/exports` 文件中读取它的共享指令。在此，用一些表示共享方式的选项来添加想要共享的目录。需要用到以下指令。

```
directory networknfs options, networknfs options
```

选择想要共享的目录、目录共享所在的网络、NFS 选项。看一个将要用到的例子。

```
/data/staff 192.168.0.2/255.255.255.255(rw,root_squash, subtree_check,fsid=0)
```

该指令要向 192.168.0.2/32 地址上的主机共享 `/data/staff` 目录。其中的 IP 地址也可以是个全局域名（FQDN）（比如 `fileserver2.example.com`）或者整个域（比如 `*.example.com`）。

下一步，设置下列选项“`rw, root_squash, subtree_check, fsid=0`”。第一个选项允许共享目录是可读和可写的。选项 `root_squash` 意味着远程主机上的 `root` 用户对于匿名的 UID/GID 有

一个 UID/GID 集合——这意味着该远程 root 用户在共享目录上丧失了 root 的权限。这是为了防止网络挂载被远程 root 用户取消。下一个是 `subtree_check` 选项，它通常代表 NFS 服务器在请求进入时检查文件系统的子树。如何设置该值取决于共享目录中存在哪种文件，其默认值是 `no_subtree_check`。打开该选项是考虑到共享目录中的文件被客户端访问之前，可能会改变和被重命名。如果是一个只读文件系统，则可以用 `no_subtree_check` 关闭该选项。

还可以设置其他诸多选项，建议读者阅读 `exports` 的手册页获得更多的详细信息。为了使上述设置生效，需要运行以下命令。

```
$ sudo exportfs -a
```

该命令假设 NFS 服务正在运行，如果没有运行，可以输入以下命令启动它。

```
$ sudo /sbin/service nfs restart
$ sudo /usr/sbin/invoke-rc.d nfs-kernel-server restart
```

当已经重启服务，定义了新的网络共享，就可以尝试在远程主机上挂载共享目录。接下来查看 NFS 共享目录是否正在服务，输入 `showmount` 命令检查 NFS 共享。

```
$ showmount -e localhost
Export list for localhost:
/data/staff 192.168.0.2/255.255.255.255
```

可以看到输出信息显示 NFS 共享目录和可连接到该共享目录的主机 IP 地址。还可以用“`showmount -e someotherhost`”命令查看另一台主机。

在远程主机上，比如 192.168.0.2，需要发出以下命令把共享目录 `/data/staff` 挂载到 `/data/remote` 目录。

```
$ sudo mount -t nfs4 -o rw,intr,hard 192.168.0.1:/data/staff /data/remote
```

这将把 `/data/staff` 目录挂载到远程主机的 `/data/remote` 目录。正如设置语句“`192.168.0.1:/data/staff`”所表示的，我们指定了共享名 `/data/staff`，然后是共享所在的主机。这里所请求挂载的目录正是远程主机上有权访问的目录，然而，如果愿意，也可以指定根目录 `/`，这样就可以挂载所有有权访问的共享目录。

下面用 `mount` 命令挂载一个 `nfs4` 类型的文件系统，并在该共享目录上设置了下列选项：读/写、可中断的（由 `intr` 所指定）以及 `hard`。第一个选项意义明显，后两个就不太明显了。NFS 传统上有个怪癖，就是如果共享文件系统的主机发生了小问题，它会严重影响所有连接到它的主机，直到服务恢复或者重启主机。`hard` 选项意味着该挂载的文件系统将被当作本地文件系统一样对待。如果它被设置为 `soft`，而不是 `hard`，一旦该文件系统闲置一段时间，它将被主动卸载。为了获取更多有关挂载选项的信息，请查看 `mount` 的手册页。

要想让共享目录在主机重启时自动挂载，则需把它添加到 `/etc/fstab` 文件中。

```
192.168.0.1:/ /data/remote nfs4 rw,hard,intr,_netdev 0 0
```

在此，访问主机 192.168.0.1 上的 NFS 文件系统，并把远程共享目录挂载到 `/data/remote` 目录下。这些选项和前面的设定相同，并且多添加了一个 `_netdev` 选项。`_netdev` 选项告诉主机在网络设备启动之前不要试图挂载该文件系统。如果没有该设置，主机将不能成功挂载该文件系统，并在失败和超出尝试次数之前等待很长时间。

到目前为止，已经探究了从主机到主机的共享文件方式，并且介绍了如何创建文件共享。假设读者的主机在重启之后就可以共享目录，那么接下来将介绍另一种共享文档和信息的方式——文档管理系统。

资源

读者可以在此找到关于 NFS 更多的内容。

- <http://nfs.sourceforge.net/nfs-howto/>
- <https://help.ubuntu.com/community/NFSv4Howto>

12.4 管 理 文 档

文件共享是企业中文档分发管理的一个重要组成部分。然而，在涉及跟踪文档和对文档进行版本控制时，文件共享具有一些不可避免的缺陷。如果不在文档的写操作和权限检查方面进行大量的管理，就不能得到精细的访问控制。而且，文件共享不能对正在被编辑的文档加锁，因此不同用户能同时访问同一个文件，并分别做更改，那么当其中一个用户把修改内容保存回共享文件时，将毁掉另一个用户之前对该文件所做的更改。当然，文件共享还是需要的，只不过我们需要一种更好的方式来管理企业运作中的文档。

文档管理系统的设计和实现可满足以下 5 个需求。

- 安全地共享文档。
- 为文档提供版本控制，让历史版本不会丢失。
- 要求检出文档，以使得两个人不能同时编辑同一份文档。
- 适应公司创建、检查和发布文档的工作流程。
- 为所有共享文档提供单一的入口点，不必管理几台文件服务器和它们的共享目录。

一个设计良好的文档管理系统，通常会有一个 Web 入口，该 Web 入口将成为文档访问的中心点。该 Web 入口点还将成为安全的内联网的一部分，因为远程分公司可以通过私有的虚拟专用网络（VPN）连接到总公司并访问它。

12.4.1 使用文档管理系统

如果企业不大，可能就不需要优秀的文档管理系统（DMS）。但是，及早对设计 DMS 投入一点思考就会免去以后的许多问题，因为随着企业规模的增大，控制文档的需求会变得更加明显。一个良好的 DMS 不仅对工作流有用，在保证文档安全方面也将有显著作用。而且，给文档添加版本控制可以使同事之间共享的数据具有更大的安全性。

12.4.2 开源 DMS KnowledgeTree

为了刻画文档管理系统的作用，我们选择一款名为 KnowledgeTree 的 DMS 产品 (<http://www.knowledgetree.com/>) 来介绍文档管理系统。KnowledgeTree 是商业和开源产品的

结晶，它有一个开源社区版本——我们将安装该版本——和一个 KnowledgeTree 公司所销售的商业版本。KnowledgeTree 的商业版本包含一些附加功能（这些功能可与微软的 Windows 桌面系统紧密结合），还有售后服务。不过，考虑到我们的目的，开源社区版本已能够满足我们的所有需求。

选择 KnowledgeTree 也是因为该产品本身是基于开源软件的。它利用了第 11 章介绍过的 Apache Web 服务器和 MySQL 数据库以及开源语言 PHP。KnowledgeTree 在 GPLv3 许可证下发行。

这一节介绍如何安装 KnowledgeTree 应用程序、如何添加用户以及如何改变 KnowledgeTree 的 SSL 证书。另外，还会介绍如何检入和检出文档。

■注：KnowledgeTree 的网站 <http://www.knowledgetree> 上有许多有用的文档。

可以从“http://www.knowledgetree.com/try-now/knowledgetree_open_source_download”上下载 KnowledgeTree 产品。需要提供一些基本信息，然后转到下载页面。

■注：KnowledgeTree 安装文件不是被我们所熟悉的压缩软件打包的，我们需要从它的出品公司下载和安装它。

在下载好 KnowledgeTree 的安装文件后，需要更改文件的权限使它可执行才能安装该软件。

```
$ sudo chmod 755 ktdms-oss-3.5.4a-linux-installer.bin
```

12.4.3 安装 KnowledgeTree

首先，必须创建一个目标目录用于安装 KnowledgeTree。正如之前提到过的，该版本的 KnowledgeTree 安装包将安装一个 Web 服务器 Apache、一个数据库服务器 MySQL 以及 KnowledgeTree 应用程序。我们习惯把系统的所有数据保存在 /data 目录下，所以也把 KnowledgeTree 的安装目录 /data/Web/KT-DMS 置于 /data 目录下。用以下命令创建该目录。

```
$ sudo mkdir -p /data/Web
```

■注：该主机将保存公司中所有的文档，它依赖可靠的硬件为公司提供高效的文件存储。

下一步，运行安装程序安装所有需要的软件。KnowledgeTree 的安装程序将加载 Apache Web 服务器和 MySQL 数据库以及 KnowledgeTree Web 应用程序。这是个界面非常友好的安装过程，它从图 12-21 所显示的启动界面开始。

下一步，需要接受其许可证协议。如图 12-22 所示，这是一个 GPLv3 许可证协议。关于 GPLv3 许可证含义的更多信息可以在 GPL 网站上的 FAQ 中找到。在图 12-22 中，接受该许可证。

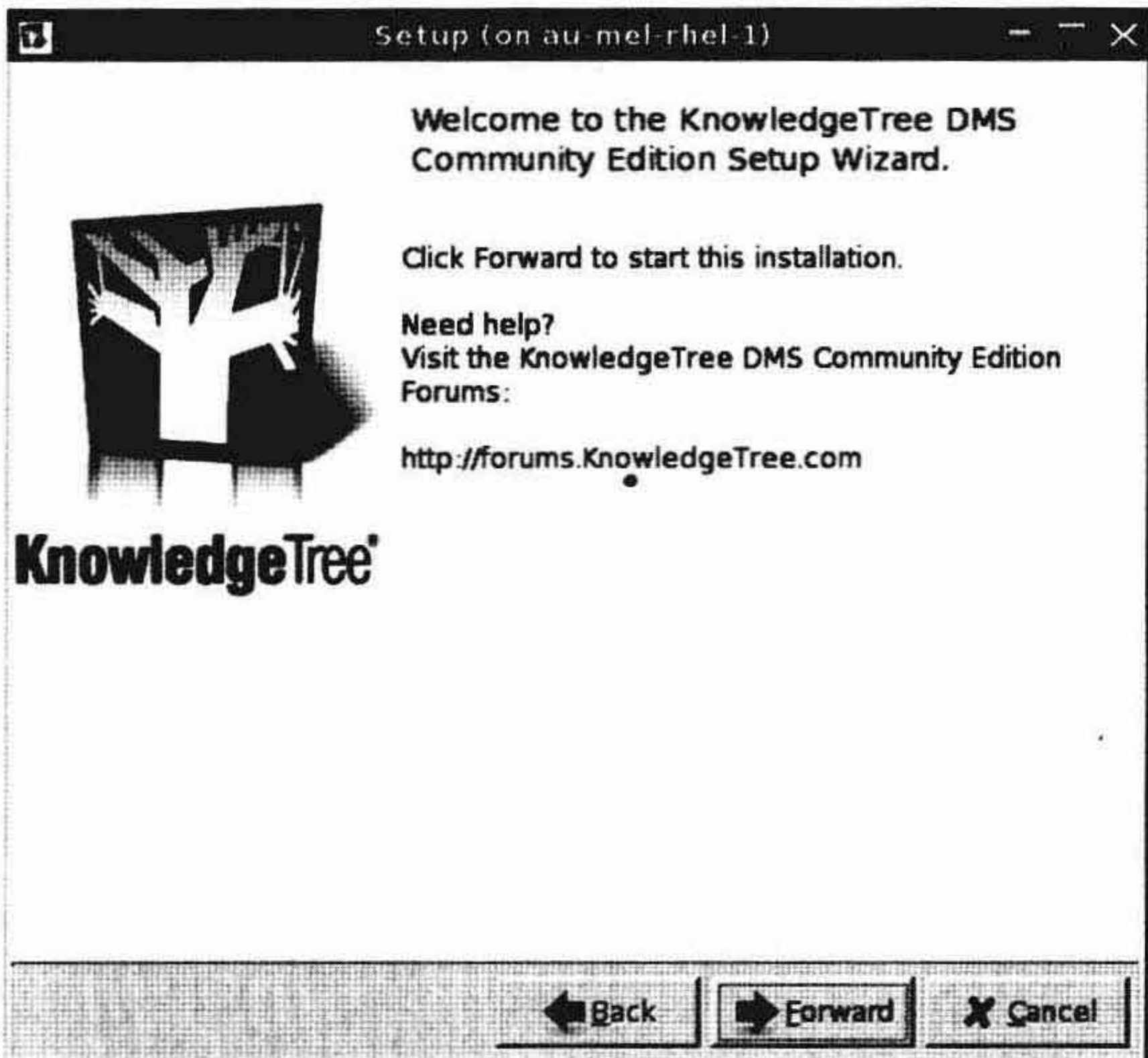


图 12-21 KnowledgeTree 欢迎界面

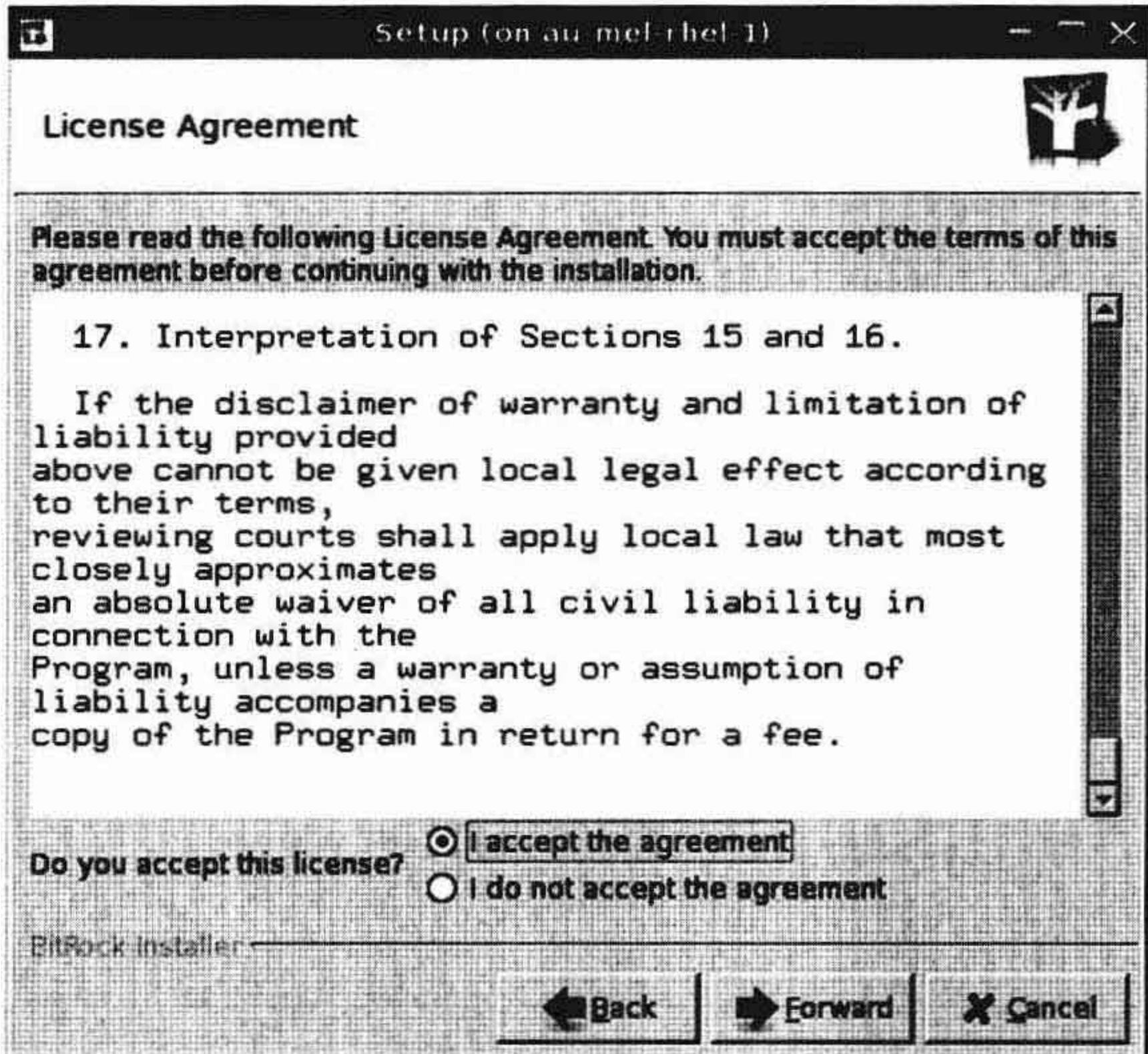


图 12-22 接受 KnowledgeTree 的许可证

下一步，需要指定应用程序的安装目录。在此指定为刚刚创建的目录“/data/Web/KT-DMS”，如图 12-23 所示。

接着需要选择 Web 服务器的端口号。选择 80 端口。如果该主机上已经运行了一个 Web 服务器，那么可以选择 8080 端口（8080 是另一个常用的 Web 服务器端口号）。在图 12-24 中，指定端口号。

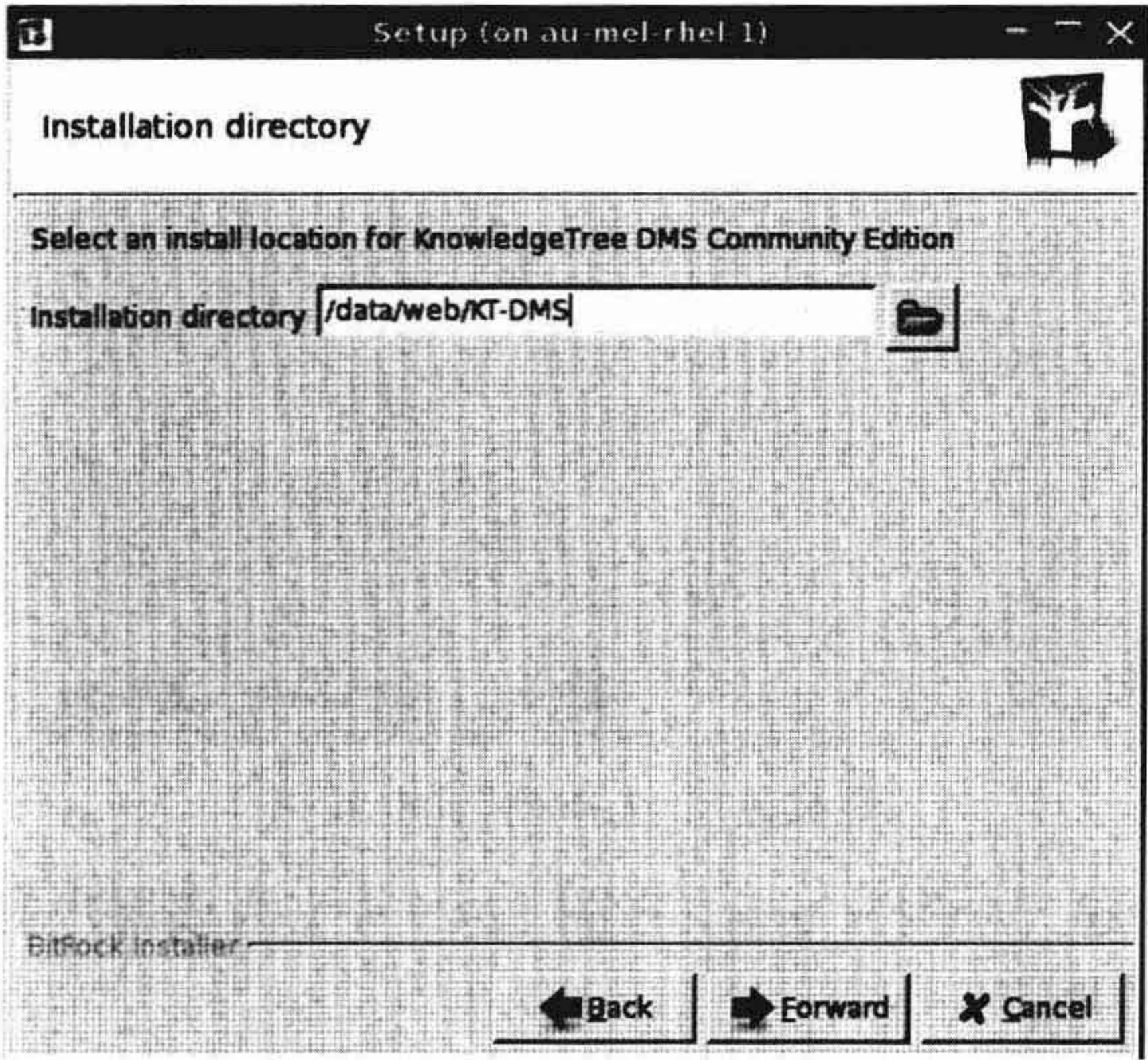


图 12-23 定义安装目录

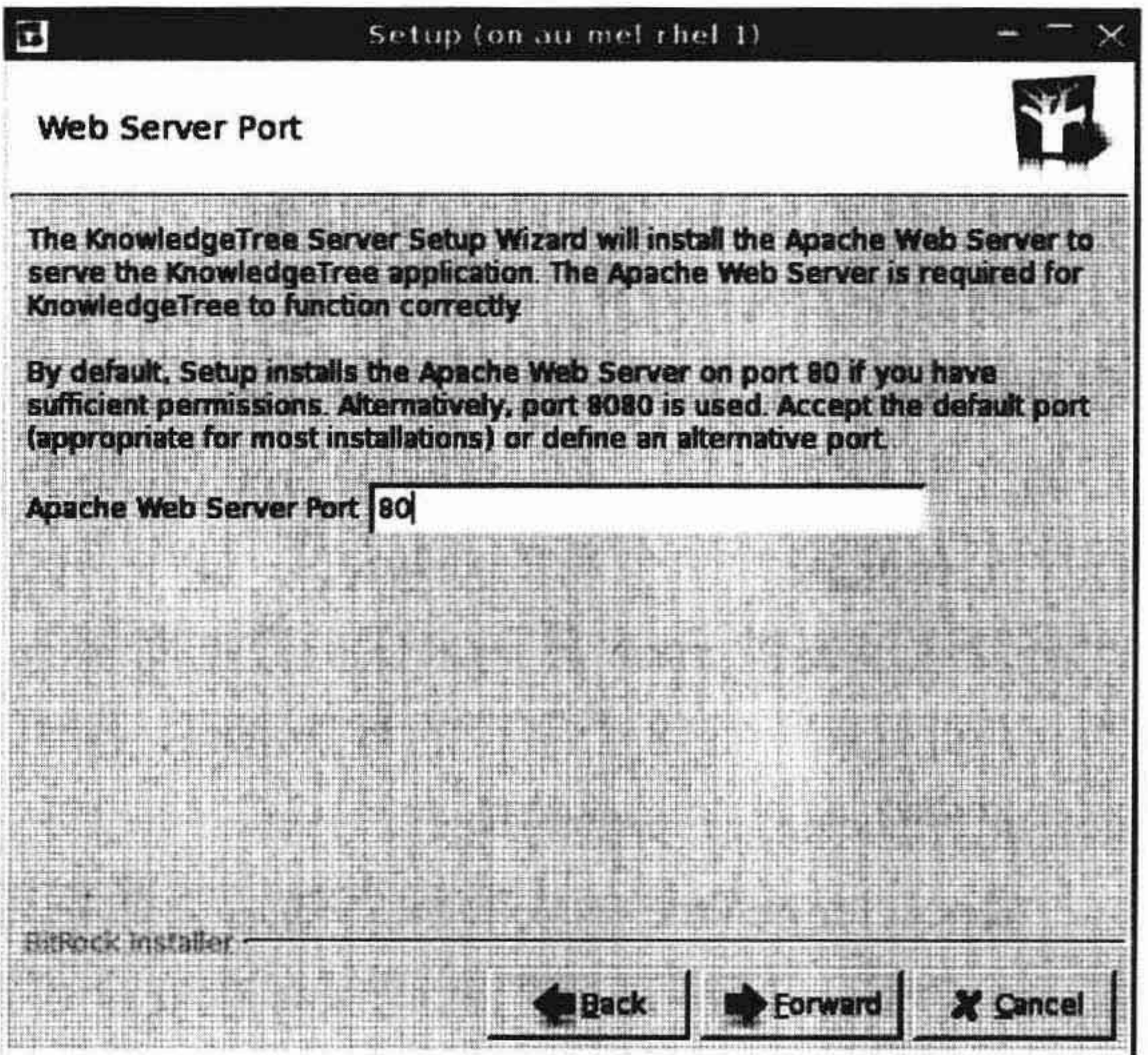


图 12-24 定义 Web 服务器端口号

下一步，选择是否想要为 Web 主机启用 SSL 支持。在图 12-25 中，明确指出启用 SSL 支持。

现在为主机生成一个 SSL 证书。在第 10 章中，读者已学会如何使用自己的认证授权创建证书。安装完成之后，我们将创建自己的证书。在图 12-26 中，需要提供主机名和该证书

的有效期。

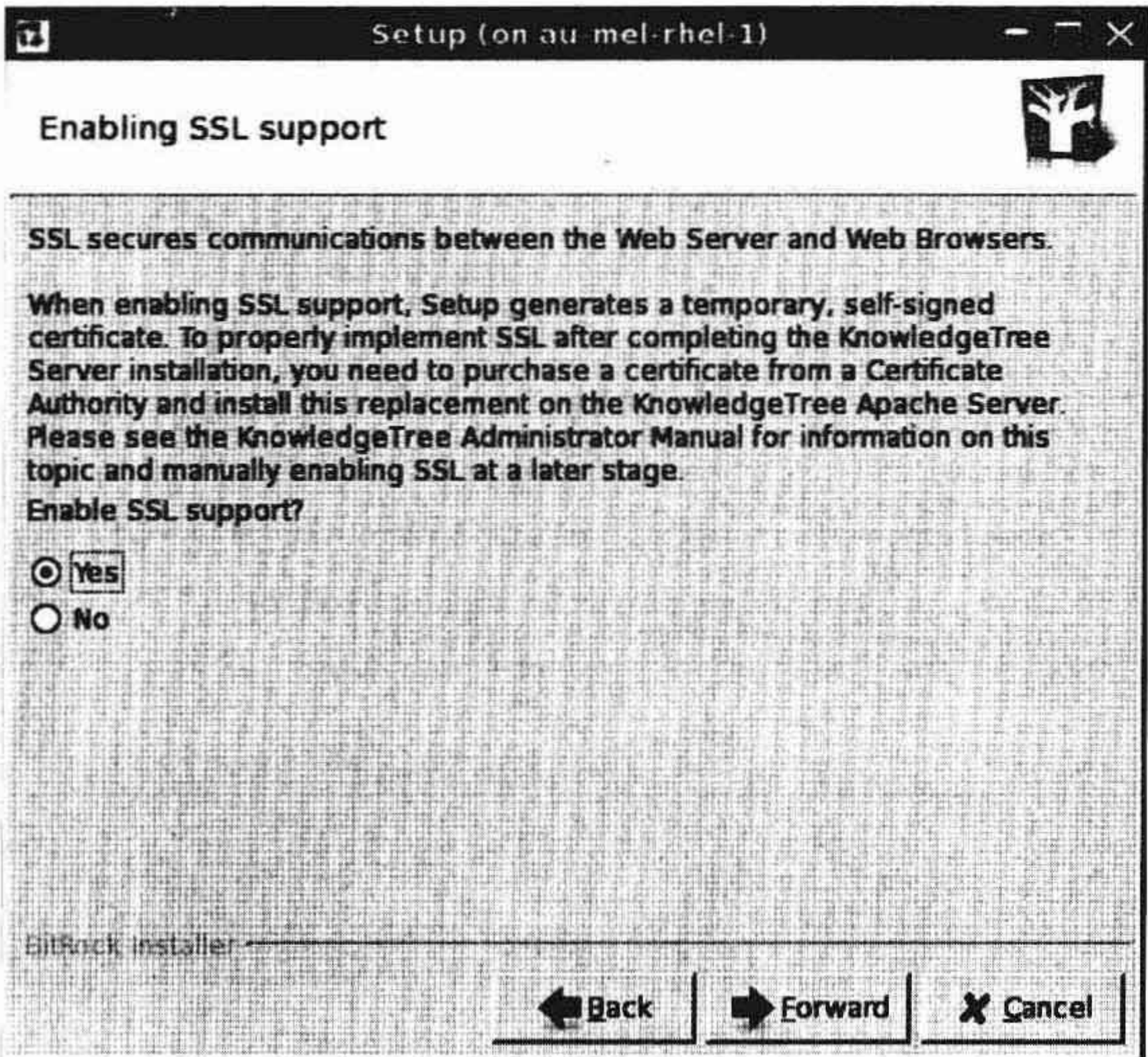


图 12-25 为 Web 服务启用 SSL 支持

这将为主机 “dms.example.com”（这里的 “dms” 代表文档管理系统）生成一个自签发的证书。这已经足够满足我们的需求了，所以没必要生成自己的证书。

下一步，选择 MySQL 服务器将要监听的端口号。同样，如果主机上已经有一个 MySQL 服务器正在运行，则必须选择一个不同的端口号，比如 3307。如图 12-27 所示，示例中使用的是标准的端口号 3306。

继续配置 MySQL，现在需要为 MySQL 数据库服务器定义 root 用户密码（见图 12-28）。正如之前提到过的，KnowledgeTree 的安装程序正在安装一个全新的 MySQL 程序，因此需要通过提供 root 密码来保护数据库服务器。在此选择一个 8 个字符的复杂密码，同时记住把密码保存在安全的地方。

KnowledgeTree 的安装程序随后为数据库创建一个用户 dms，在图 12-29 中，为该用户提供一个密码。同样需要把密码保存在安全可靠的地方。

这将为用户 dms 生成一个密码，让他能访问刚创建的数据库 dms。

现在，选择是否想要接收 KnowledgeTree 的通知信息，选择 “Yes” 或者 “No” 并且单击 “Forward” 按钮。

下一步是终端用户许可协议（End User License Agreement）。如果对它没有不满意的地方，请接受它并继续安装。如果不接受该许可，安装程序将退出。再一次，单击 “Forward” 按钮继续。

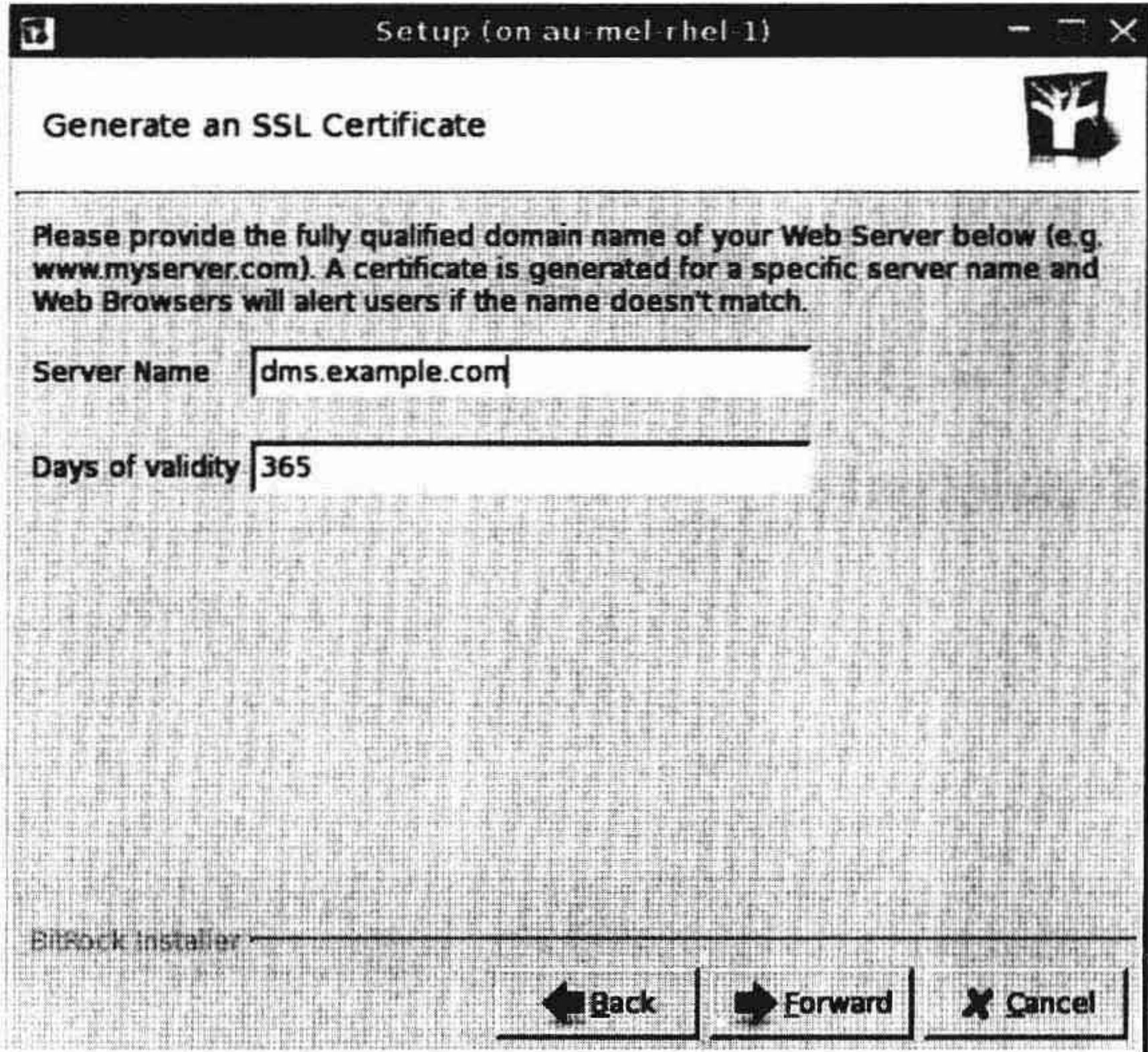


图 12-26 创建一个自签发的证书

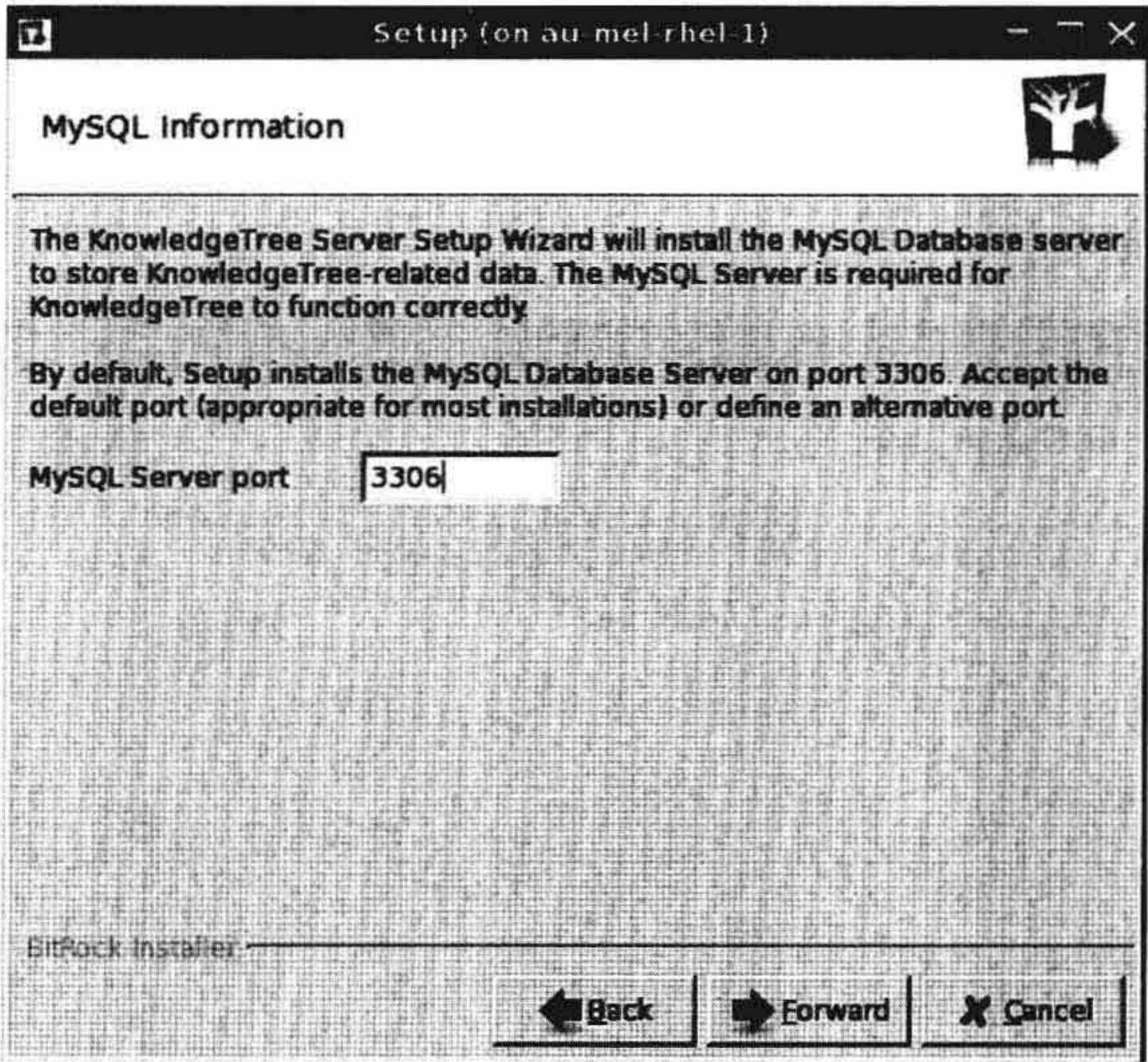


图 12-27 选择 MySQL 端口

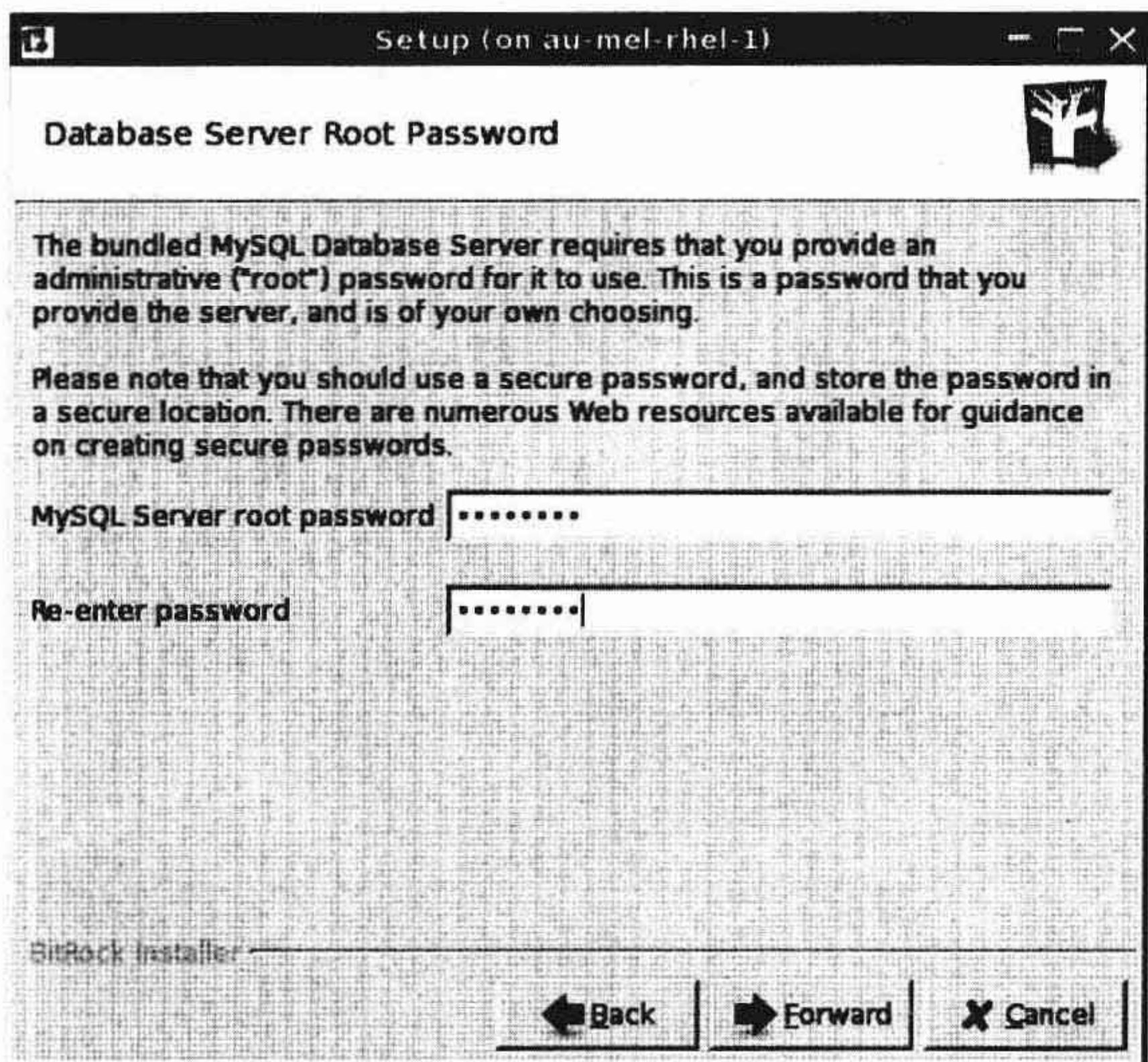


图 12-28 设置 MySQL 密码

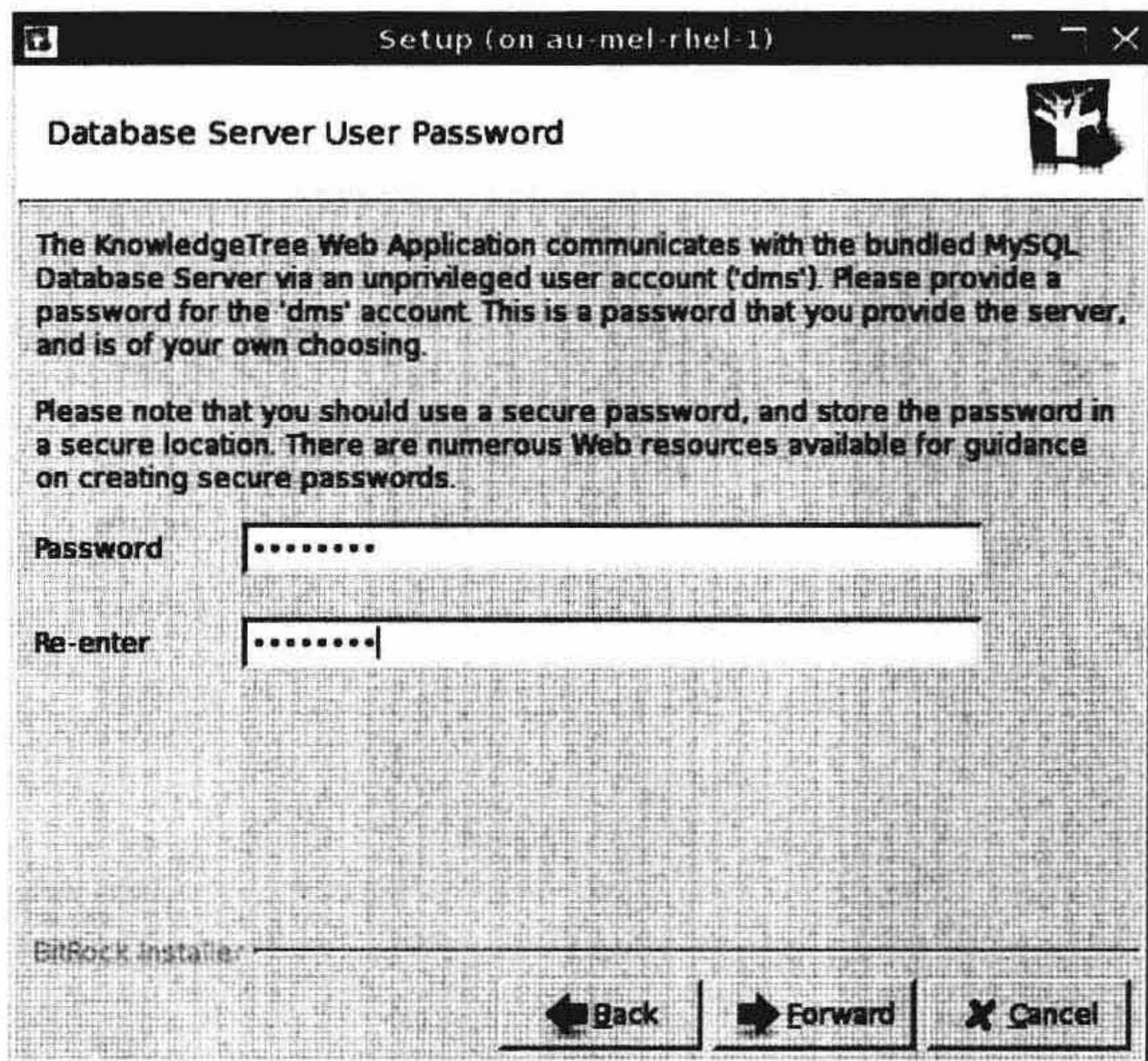


图 12-29 数据库服务器用户密码

下一步，选择是否想要安装 Windows 的拖动功能。Windows 的拖动功能允许用户从桌面窗口往 KnowledgeTree 拖动文档。如图 12-30 所示，我们拒绝了该功能，但读者需要根据自己的需求做出选择。

到此，开始了真正的安装过程。选择“Forward”继续安装。

当看到准备安装画面，再次单击“Forward”按钮开始安装，下一步，将会看到安装进度。

一旦安装完成，就可以登录到该新 DMS，如图 12-31 所示。

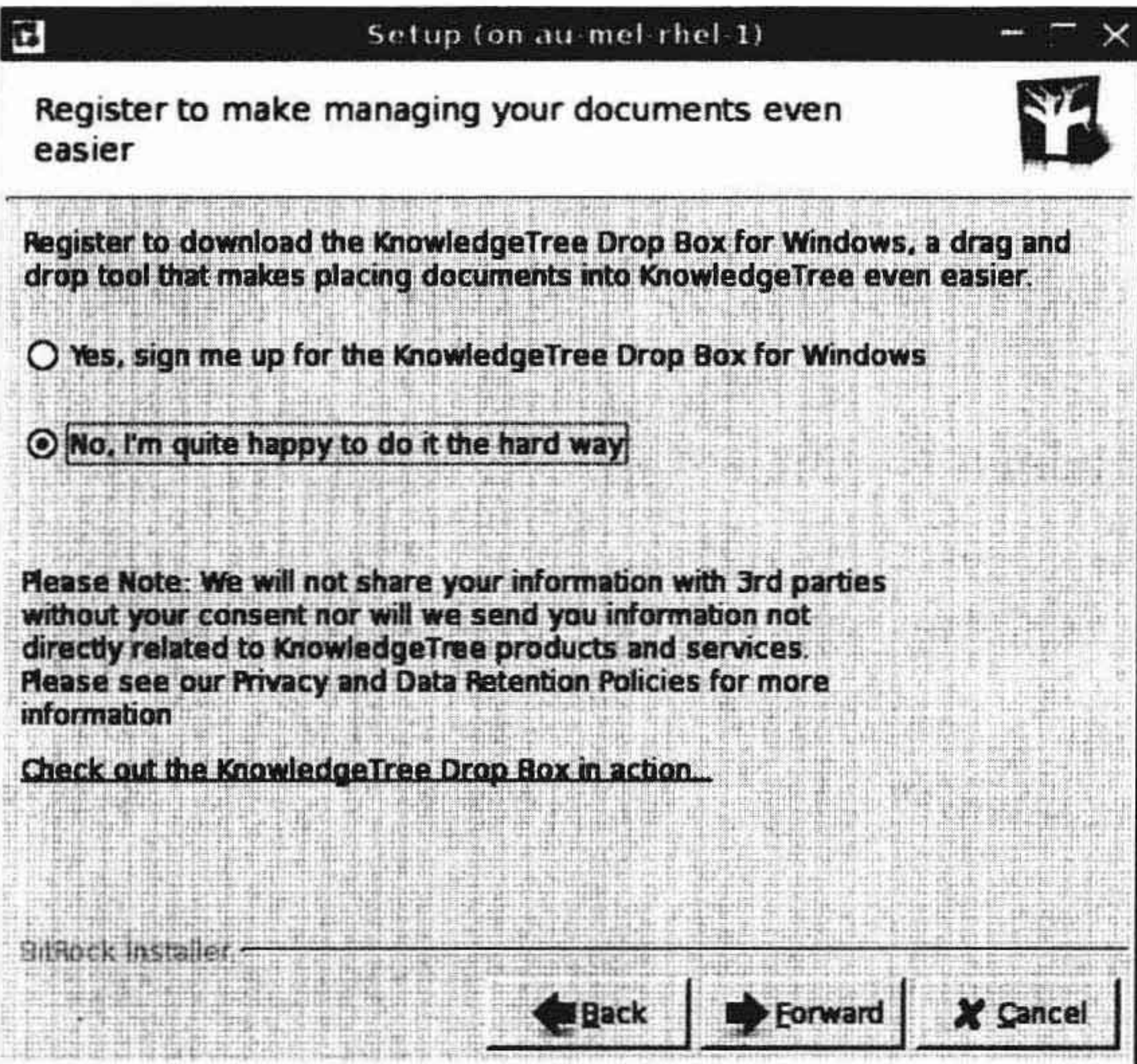


图 12-30 拒绝安装 Windows 的拖动功能

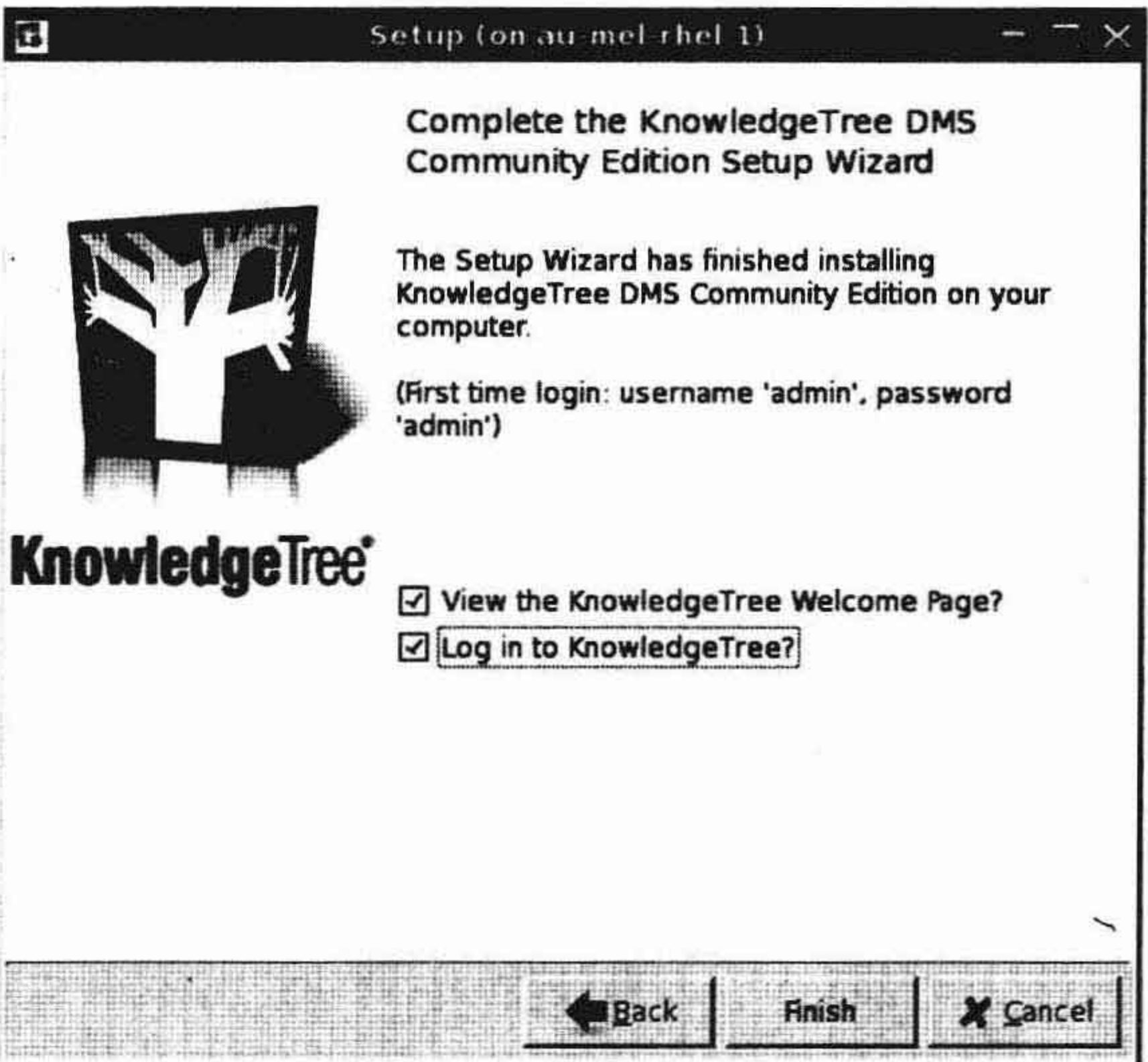


图 12-31 成功。安装完成

单击“Finish”按钮完成安装并运行浏览器。浏览器将显示 KnowledgeTree 的登录界面，如图 12-32 所示。

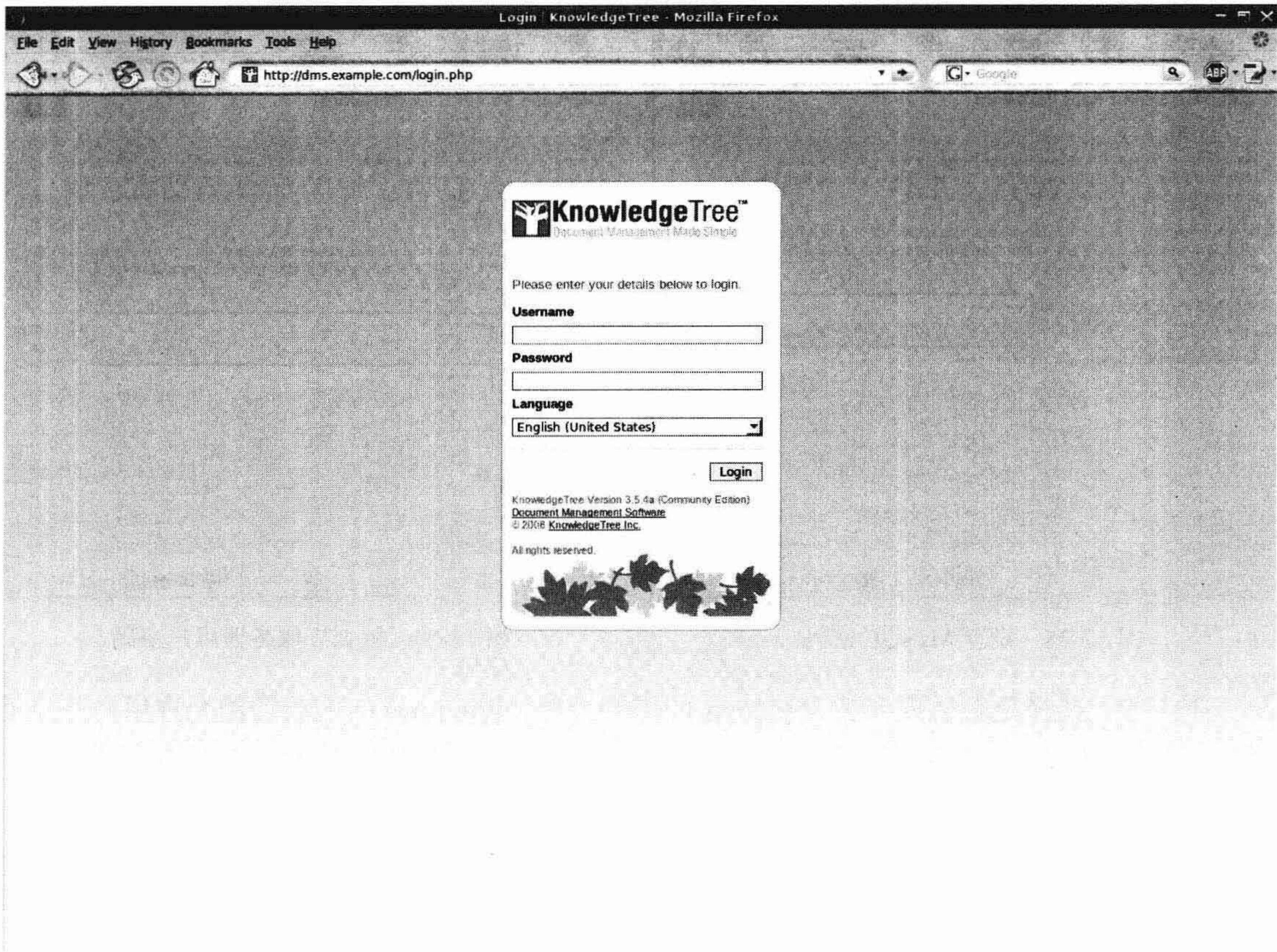


图 12-32 KnowledgeTree 登录界面

12.4.4 管理 KnowledgeTree

现在，开始管理 DMS 应用程序。可以更改管理员密码、添加用户以及建立 DMS 基本配置。

KnowledgeTree 的默认用户名和密码分别是 admin 和 admin（稍后更改它们）。当登录成功，将看到图 12-33 中的界面显示。

为了更改 admin 用户的密码，在登录之后，需要单击右上角的“Preferences”链接转到图 12-34 所示的窗口。在该窗口，可以修改 admin 用户的密码并设置其 E-mail 地址。

在图 12-34 中，还可以看到“Change your password”链接。单击该链接，显示一个熟悉的更改密码界面（见图 12-35）。

下一步，可以通过单击 DMS 主菜单上的“Administration”链接进行全局的控制管理。这里可以通过设定 workflow、文档存储以及用户和组来控制该 DMS 将如何工作，如图 12-36 所示。

使用“Users and Groups”链接，可以为 DMS 创建用户和组。在此，如果有 LDAP 或者 AD 服务器，还可以为用户认证和密码鉴定而把 DMS 绑定到已有的 LDAP 或者 AD 服务器。

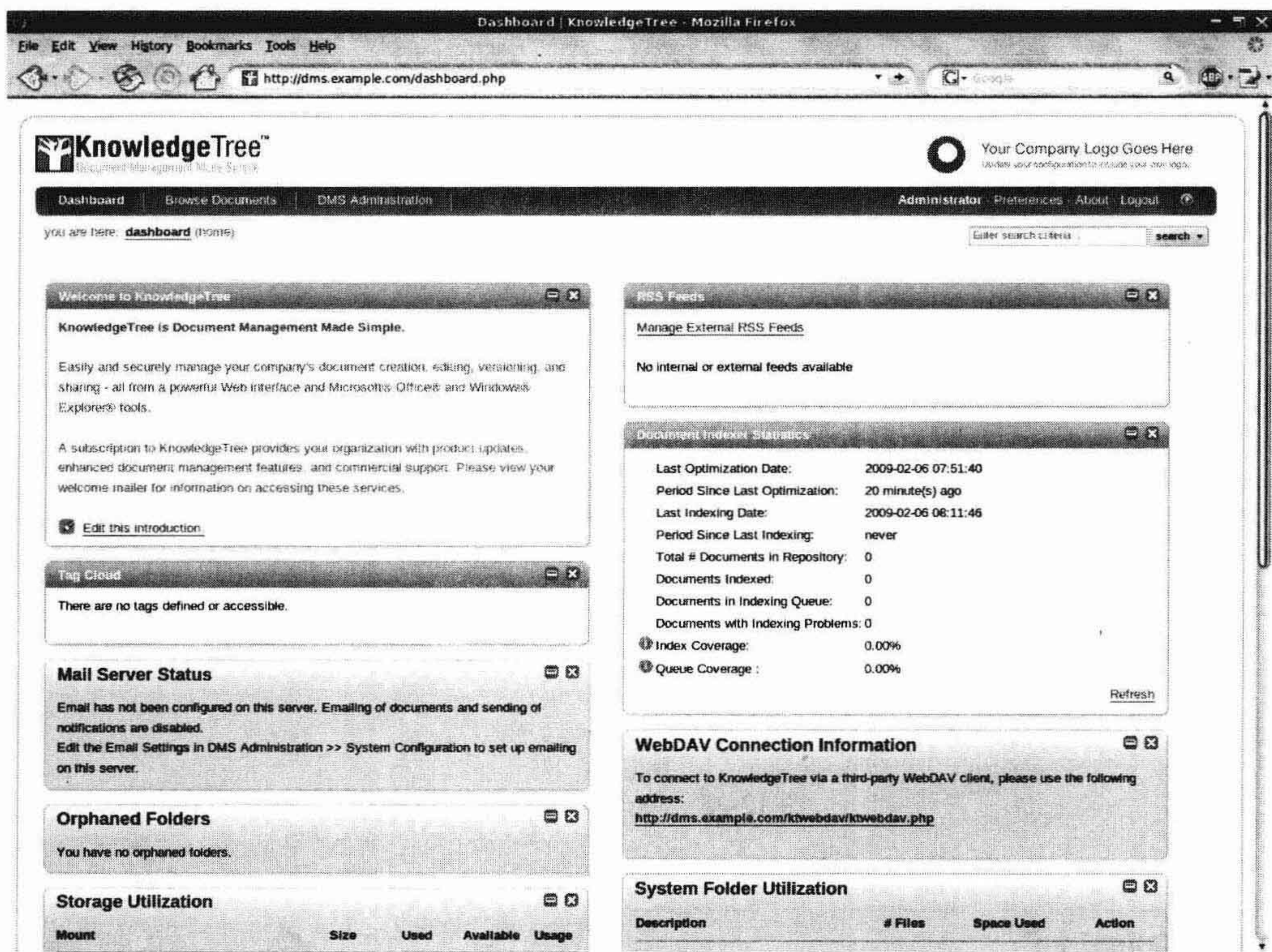


图 12-33 首次进入 KnowledgeTree

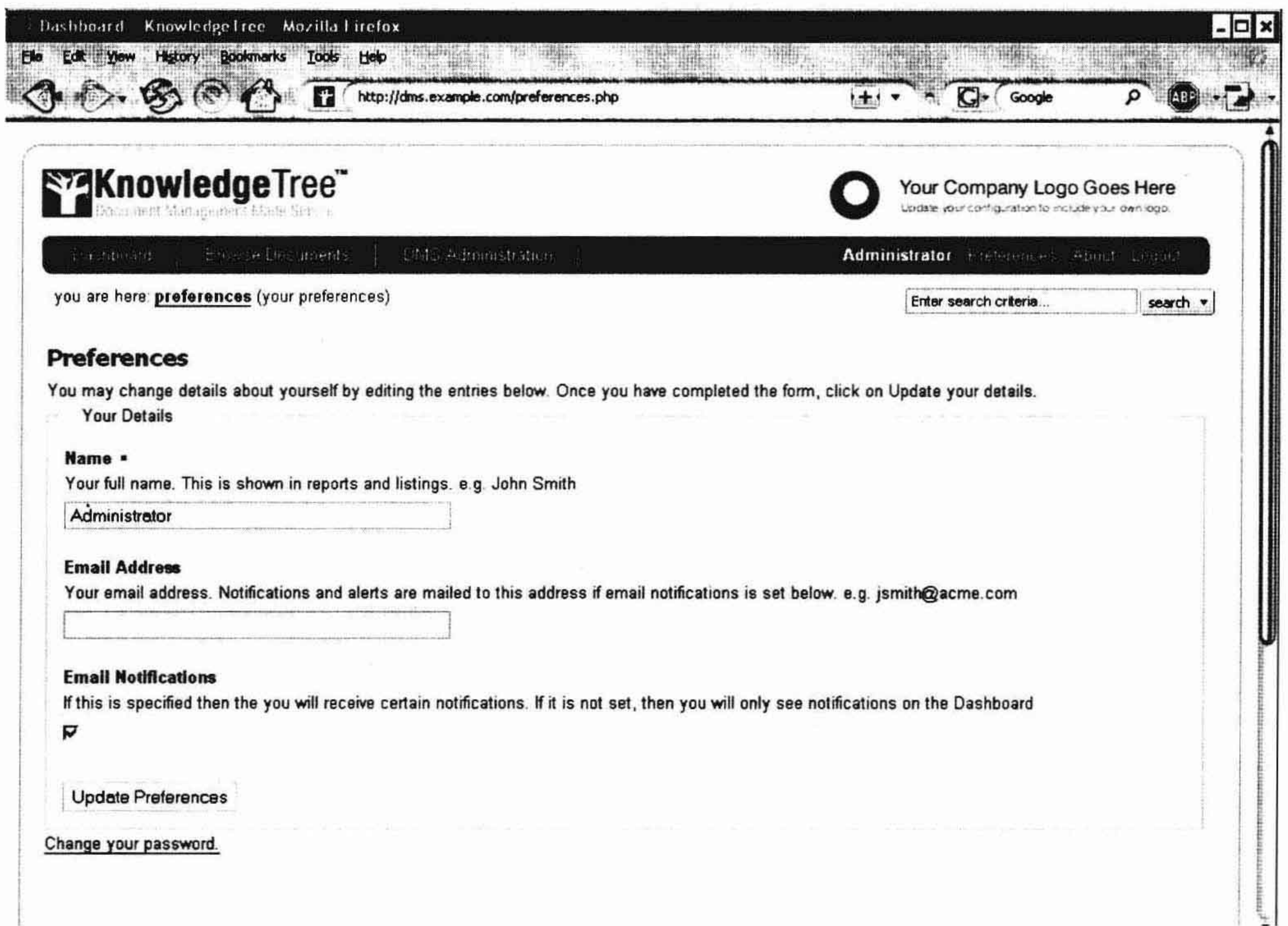


图 12-34 设置管理员信息

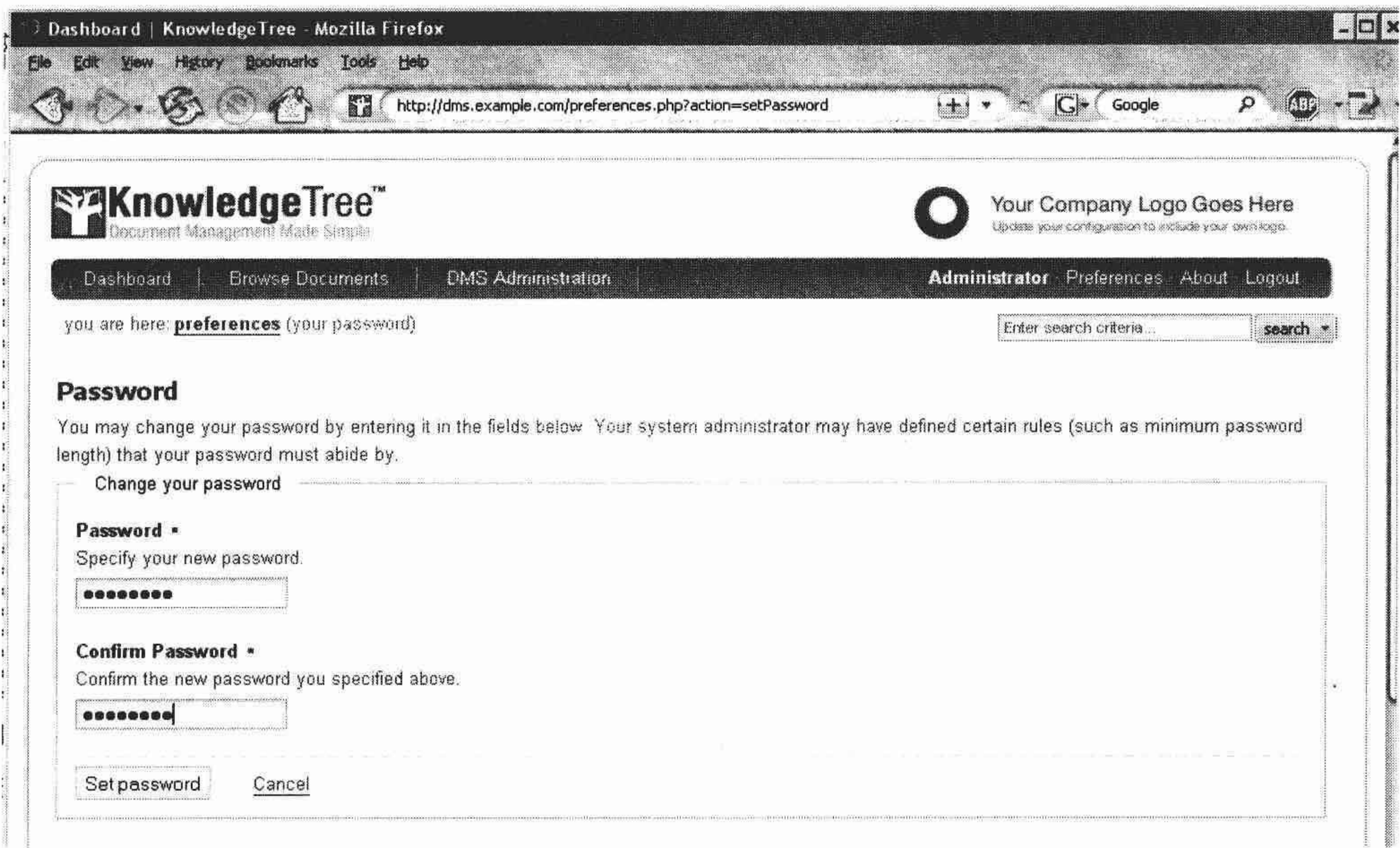


图 12-35 修改管理员密码

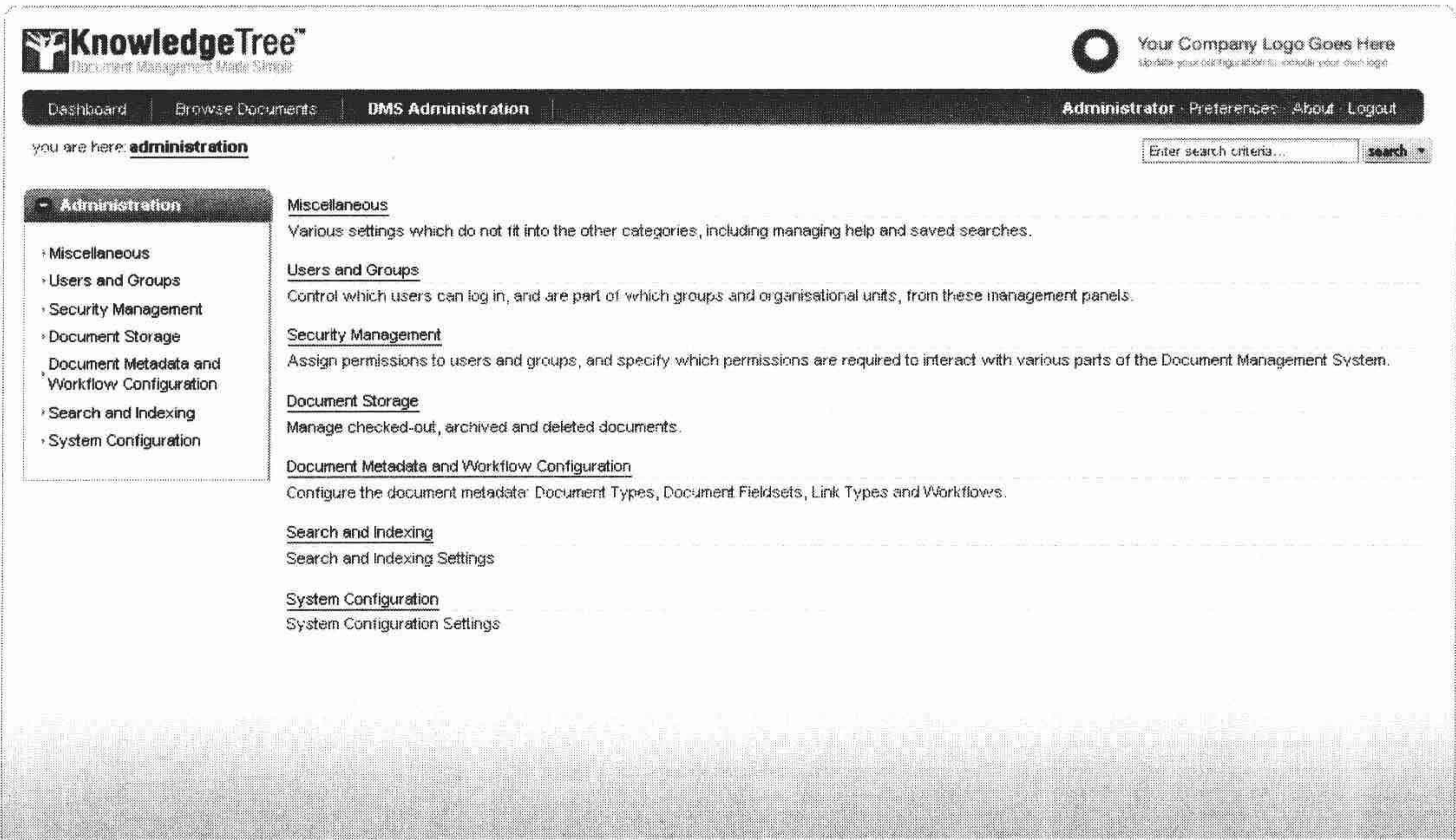


图 12-36 DMS 管理标签页

■注：第 16 章介绍 LDAP 服务器。

如果有一个外部的认证源，可以单击“User Management”链接并在先前的窗口中选择“Add user from source”来添加用户。或者，如果没有 LDAP 或者 AD 服务器，则可以再次进

入用户管理窗口手工地添加用户。图 12-37 在显示了 “Add a new user” 链接的同时，还可以看到已经添加的一些用户。

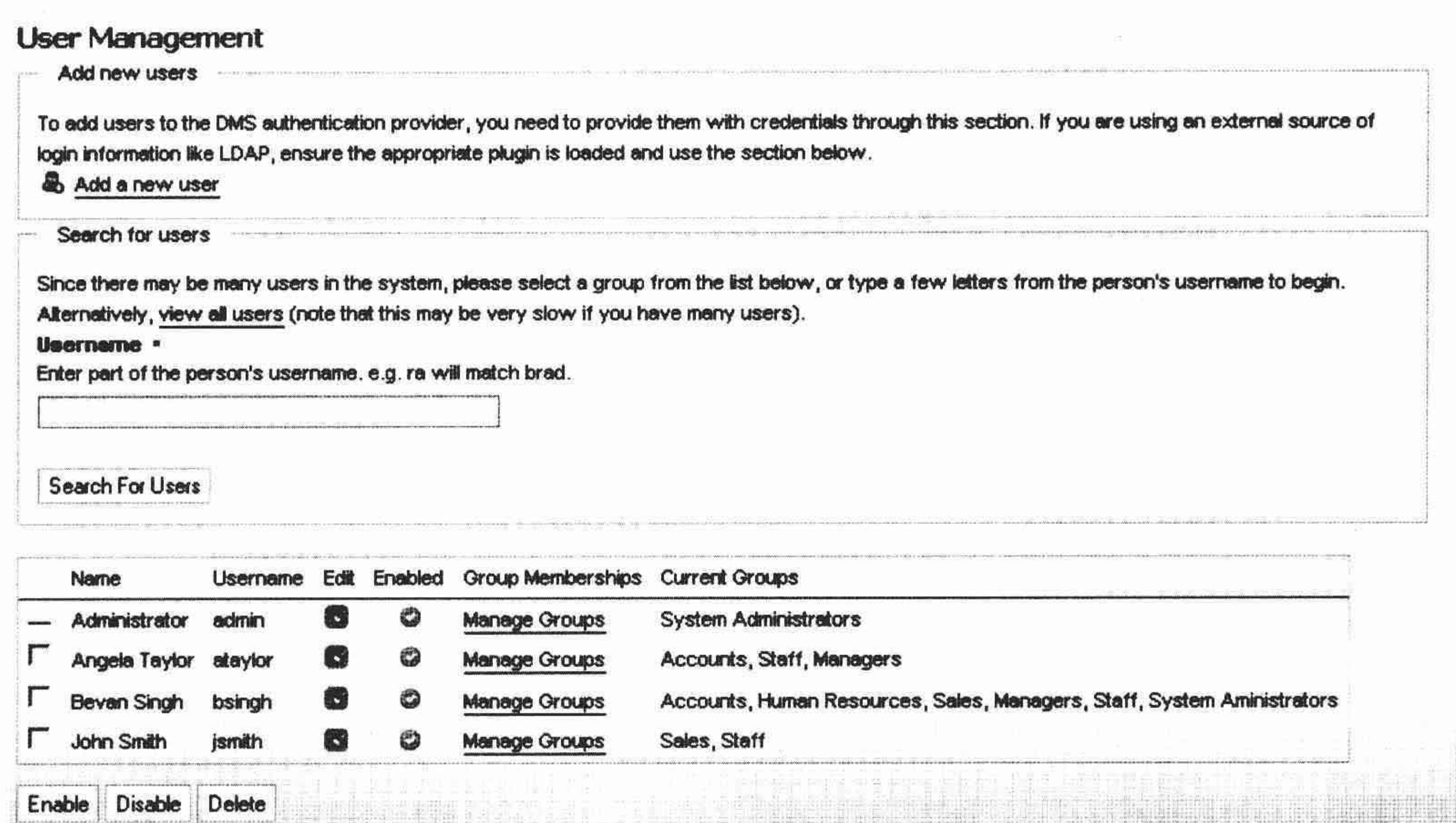


图 12-37 KnowledgeTree 的用户管理窗口

现在已经创建好了用户账号，接下来开始对 DMS 做些设置以满足公司的需求。我们并不需要 DMS 的所有功能，但有必要进一步了解用户身份、权限和工作流这些内容。如何设置 DMS 实际上需要考虑到各个公司自身的决策。在正式使用 DMS 之前，强烈建议读者阅读 KnowledgeTree 的文档，尤其是 http://docs.knowledgetree.com/manuals/ag/organizational_hierarchy.html 上的内容。

12.4.5 处理文档

接下来向读者简要介绍如何添加和检出文档。示范的 DMS 站点已经有一些目录了，它们都是刚创建的。我们打算把公司将来的所有文档都放在 example.com 目录下，而且每当有了新的业务，就把新业务的所有文档放在业务自身的目录下。同样地，把所有与其他企业相关的敏感文档存放在一个叫做 Executive 的目录下。

现在开始介绍，首先选择导航菜单上的 “Browse Documents” 菜单，如图 12-38 所示。

进入 “example.com” > “Sales” 目录，把桌面系统上的一个文件添加到 Sales 目录。在左侧的菜单中，可以看到一个很大的 “Upload Document” 按钮。单击该按钮，显示如图 12-39 所示的文档上传页面。

添加一个名叫 forward_order_big_client.doc 的桌面系统文档。如果需要，可以给它起一个不同的名字。选择文档类型为 Sales。文档类型是个可配置项，它持有所存储文档的元数据。元数据用来在 DMS 内部组织文档和查找文档。可以在 DMS 的管理标签页中创建文档类型。

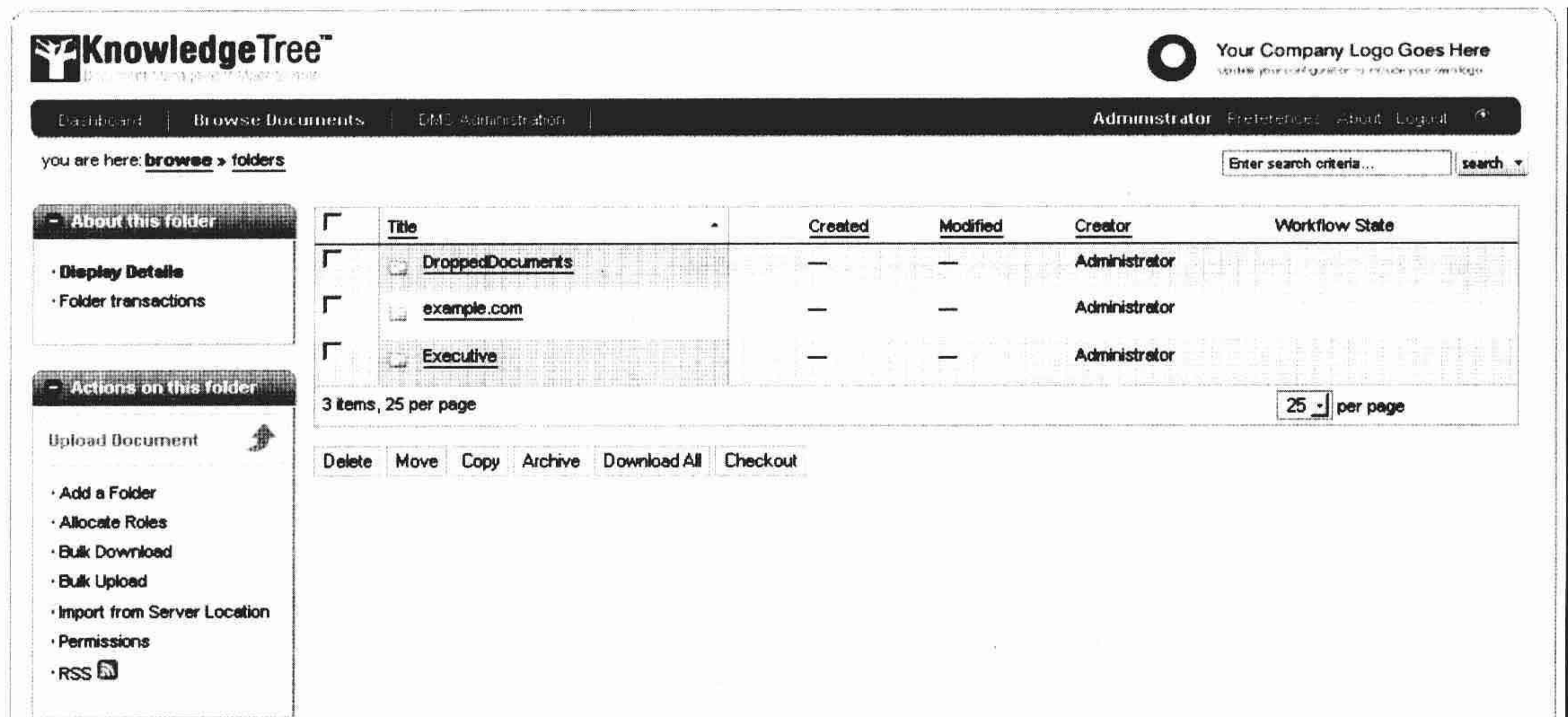


图 12-38 浏览文档

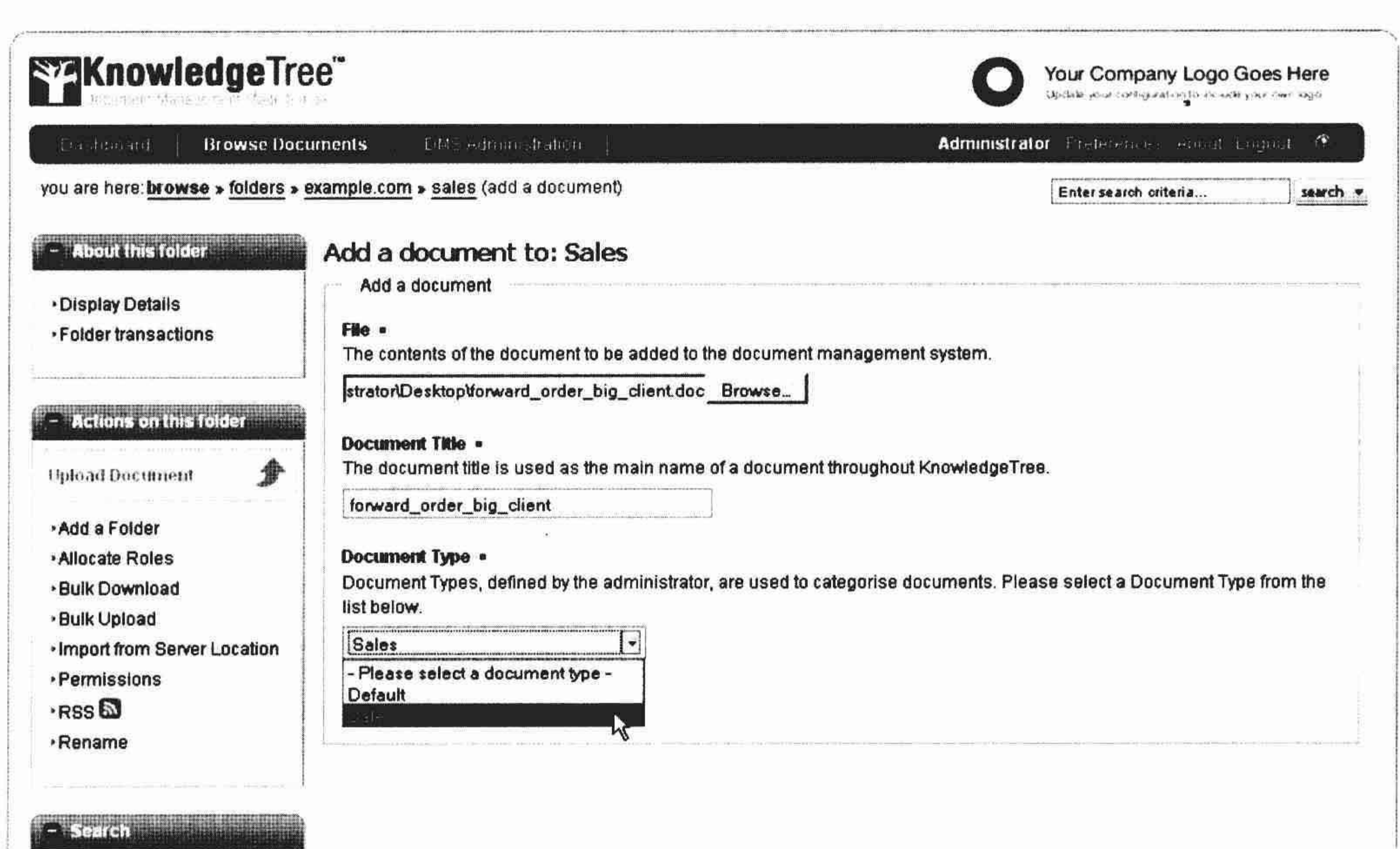


图 12-39 上传文档到 Sales 目录

注：可以在“http://docs.knowledgetree.com/manuals/ag/creating_effective_document_ty.html”上阅读到关于创建文档类型的更多信息。

在图 12-40 中，可以看到文档类型的元数据。可以把它们作为详细信息添加到文档。有些元数据是必填的，有些是可选的。红色星号表示的是必填项，当配置文档类型的时候需要对此做设置。

Specify Metadata

Tag Cloud

Tag Cloud

Tag

Tag Words

General information

General document information

Document Author

Please add a document author

Category

Please select a category

No selection.

Media Type

Please select a media type

No selection.

Sales

Sales document

Orders

Orders document

Client

Customer Name

Save Document

图 12-40 为文档添加元数据

保存文档，进入 Sales 目录，现在可以在该目录中看到保存后的文档，如图 12-41 所示。

you are here: [browse](#) » [folders](#) » [example.com](#) » [sales](#)

Enter search criteria... search

About this folder

Display Details

Folder transactions

Actions on this folder

Upload Document

Title	Created	Modified	Creator	Workflow State
<input type="checkbox"/> Sales Managers	—	—	Administrator	—
<input checked="" type="checkbox"/> forward_order_big_client (24Kb)	2009-02-13 14:20	2009-02-13 14:20	Administrator	—

2 items, 25 per page

25 per page

Delete Move Copy Archive Download All Checkout

图 12-41 该文档现已在 DMS 中

现在介绍如何检出文档。在文档的左边有个可供选择的复选框。当选中复选框，就能执行以下某些按钮的功能。这些按钮有“Delete”、“Move”、“Copy”、“Archive”、“Download All”和“Checkout”。

下面将检出文档 forward_order_big_client.doc，并做些改动，然后把它存回 DMS。通过选中复选框来选择文档 forward_order_big_client.doc，单击“Checkout”按钮。在图 12-42 中，可以看到我们需要选择是否想要继续检出该文档。

单击“Continue”，会看到另一个页面，该页面

you are here: [bulk actions](#) » [checkout](#)

Checkout

The action will be performed on the following documents and folders:

Documents

• forward_order_big_client

The action can be performed on the entire selection.

Continue Cancel

图 12-42 检出文档

要求给出一个检出文档的原因。检出原因将被记录在文档旁边。作为管理员，可以利用这一信息来查看哪些人何时检出文档以及为什么检出文档。这是一个需要大家遵守的审计跟踪制度，或者是一种调查什么人什么时候干了什么事的方法。

在图 12-43 中，用一个简单的理由检出该文档。

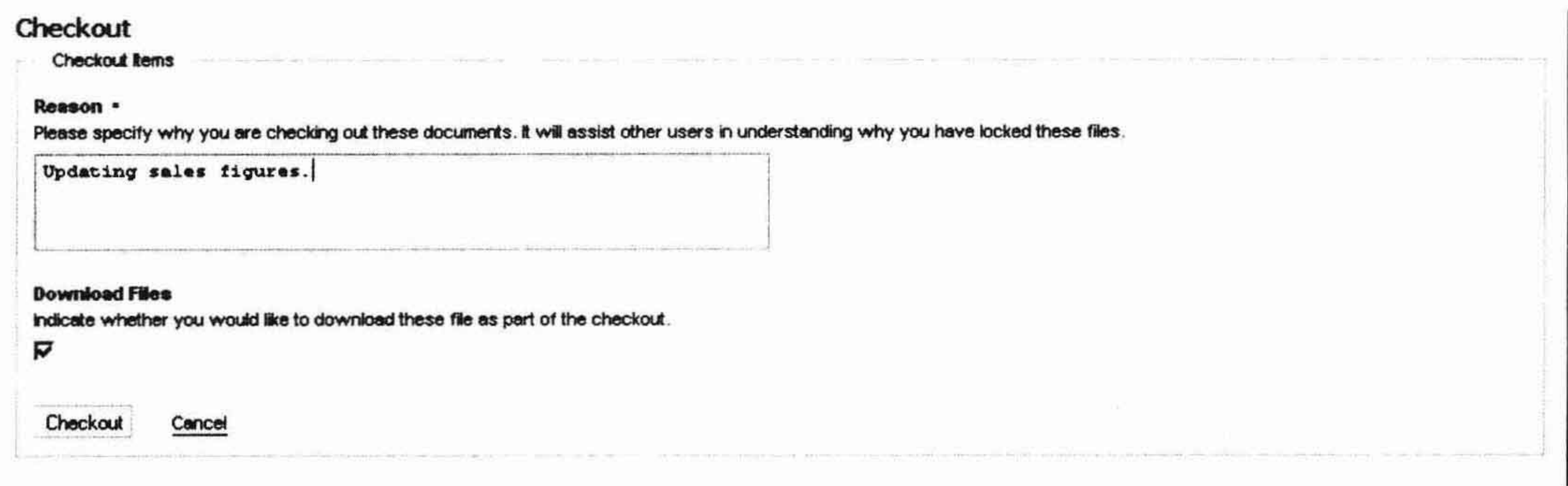


图 12-43 为文档添加审计跟踪

现在得到该文档的一个压缩版本（把文件压缩后下载有利用提高大文件的下载速度）。我们可以把该压缩版本保存到相应的目录，然后直接解压缩，也可以根据压缩软件来选择目录，把该文档解压缩到适当的目录，然后开始编辑文档。当把文件解压缩后，该文档将处于和 DMS 相同的目录结构下。所以本例中的文档可以在“根目录” > “example.com” > “Sales” 目录下找到。在此对该文档做了一点小小的改动，然后把它放到桌面上。

当浏览回到 Sales 目录，将会在此发现 forward_order_big_client.doc 文档。如果单击该文档，右边的菜单将会改变，显示出检入文档的选项。在图 12-44 中，可以看到菜单改变了，还会看到该文档的一些详细信息，包括在文档信息后的“**This document is currently checked out by you**” 该信息语句。

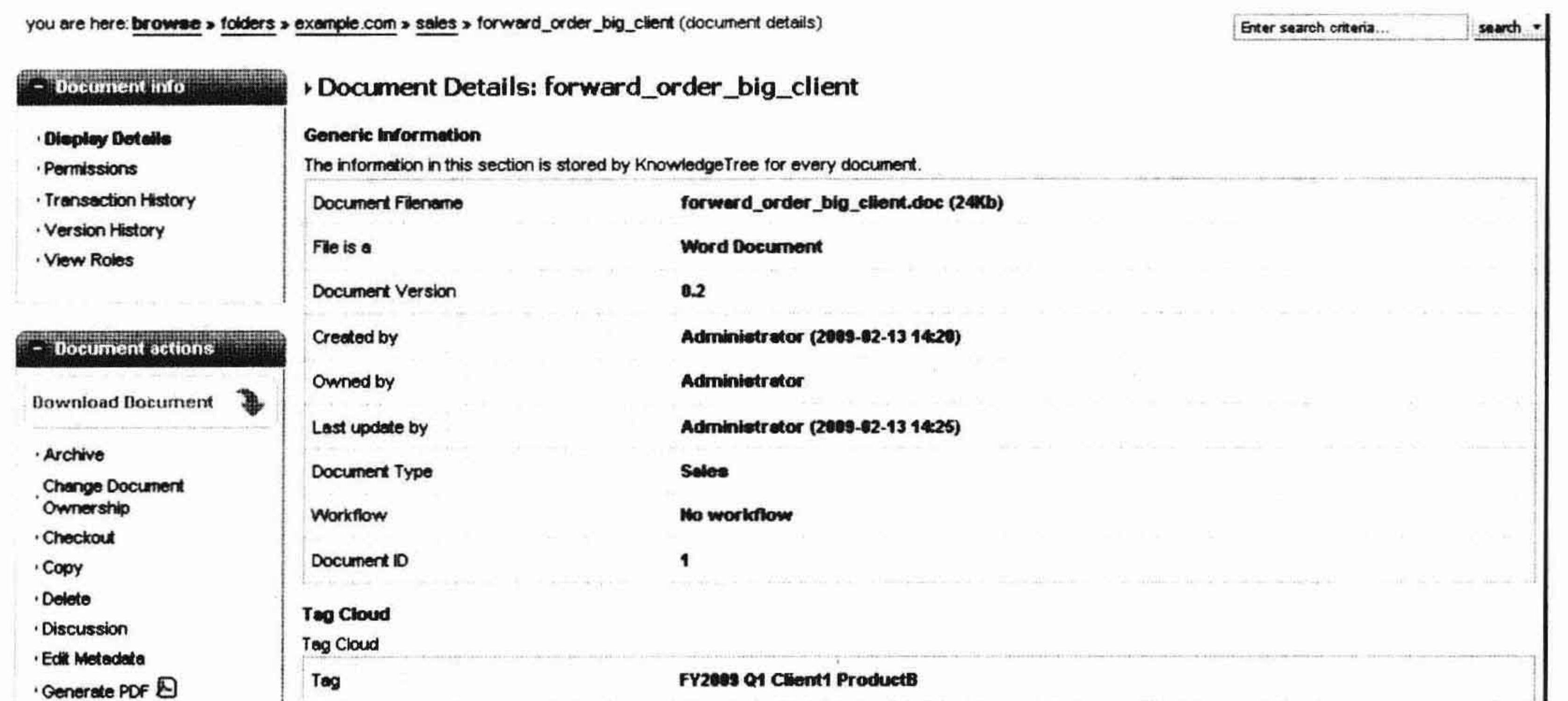


图 12-44 已检出文档的详细信息

单击“Check in Document”按钮，再次打开文件浏览对话框，检入该文档（在我们的例

子中，该文档是 forward_order_big_client.doc)。需要选择这次更新是主版本更新还是次版本更新，并且填写必填的“Reason”文本框来解释修改的原因，如图 12-45 所示。

Checkin Document:
forward_order_big_client

Checking in a document updates the document and allows others to make changes to the document and its metadata.
If you do not intend to change the document, or you do not wish to prevent others from changing the document, you should rather use the action menu to cancel this checkout.

Checkin Document

File *

Please specify the file you wish to upload. Unless you also indicate that you are changing its filename (see "Force Original Filename" below), this will need to be called forward_order_big_client.doc

Administrator\Desktop\forward_order_big_client.doc

Browse...

Major Update

If this is checked, then the document's version number will be increased to 1.0. Otherwise, it will be considered a minor update, and the version number will be 0.2.

☐

Reason *

Please describe the changes you made to the document. Bear in mind that you can use a maximum of 250 characters.

Updated sales figures

Force Original Filename

If this is checked, the uploaded document must have the same filename as the original: forward_order_big_client.doc

☒

Checkin

Cancel

图 12-45 把文档检入回 DMS

现在，再次单击该文档，将会看到版本号已经增加了。也可以使用“Document info”中的菜单选项获得关于该文档的更多信息，比如，单击“Version History”菜单显示该文档版本的详细信息，和在图 12-46 中看到该文档的版本信息一样。

Document info

- Display Details
- Permissions
- Transaction History
- Version History
- View Roles

Document Version History:
forward_order_big_client

This page lists versions of document metadata and allows you to compare a metadata version with the current metadata content.

User	Metadata Version	Content Version	Compare with Current	Compare with Other Version	Date Created
Administrator	1	0.2	current version	Metadata	2009-02-13 14:25:59
Administrator	0	0.1	Metadata	Metadata	2009-02-13 14:20:27

图 12-46 文档版本信息

可以在该页面上查看文档在 0.1 版或者 0.2 版的时候是什么样子，也可以比较版本之间的元数据。如果在 0.2 版的文件上做了点修改，还可以方便地把它回滚到 0.1 版。

还可以用“Document info”中的菜单更改文件权限，查看关联该文档的用户角色。也可以用它来查看事务历史，了解在该文档的生命周期内有哪些人检入和检出该文档。

DMS 系统的优点显而易见。我们能为 DMS 中的文档设置细粒度的访问权限，也能在其存在周期中查看它们是如何被修改的，还能处理那些比我们在这介绍的更加复杂的情况。如果需要，可以访问 KnowledgeTree 的网站，观看那里的一些网络直播，学习该产品的其他功能 (<http://www.knowledgetree.com/resources>)。

12.4.6 启动和停止 KnowledgeTree 文档管理系统

启动和停止 KnowledgeTree DMS 是通过执行 `dmsctl.sh` 脚本完成的。`dmsctl.sh` 脚本位于 KnowledgeTree 的根目录中。在本例中，`dmsctl.sh` 脚本位于目录 `/data/Web/KT-DMS`。该脚本接受 `start`、`stop`、`restart` 和 `help` 等参数。

`dmsctl.sh` 脚本实际上是一个启动脚本。在 Red Hat 主机上，可以通过 `chkconfig` 命令把它添加到系统的自启动组。

```
$ sudo cp /data/Web/KT-DMS/dmsctl.sh /etc/init.d/kt-dms
$ sudo chkconfig kt-dms on
```

Ubuntu 上相应的命令是 `update-rc.d`。

```
$ sudo update-rc.d kt-dms defaults
```

12.4.7 带 SSL 的安全 KnowledgeTree

如果读者一直按照我们的做法安装和操作 KnowledgeTree 系统，那么读者的 KnowledgeTree 网站目前正处于未加密的运行状态。在此建议为 KnowledgeTree 网站添加 SSL 支持，保护密码和文档这些敏感信息的安全。为了给网站添加 SSL 支持，必须更改 KnowledgeTree 的 Apache 服务器配置，以便 Apache 用自己的 SSL 证书和密钥。下面将做些示范。

我们在第 10 章创建了一个 CA (certificate authority) 服务器来签署自己的 SSL 证书。接下来将创建一个私钥和一个证书请求文件，然后用 CA 签署它。

```
$ sudo openssl req -new -newkey rsa:2048 -nodes -keyout dms.example.com.key ➡
-out dms.example.com.req
```

把请求文件复制到 `/etc/CA` 目录，然后用 CA 签署该请求文件。

```
$ cd /etc/CA
$ sudo openssl ca -out dms.example.com.cert -config ./openssl.cnf -infiles ➡
dms.example.com.req
```

现在分别把 `dms.example.cert` 和 `dms.example.key` 文件复制到 `/etc/pki/tls/certs` 和 `/etc/pki/tls/private` 目录下。即使以后不会直接在这些目录中使用它们，这些目录依然是放置证书文件的合适地方。`dms.example.com.key` 文件需要有 600 安全权限，而且该许可证必须可读，所以设置它的权限为 644。

与其更改 Apache 的配置文件 `/data/Web/KT-DMS/apache2/conf/ssl.conf`，不如更改该配置文件所指向的文件，相关的文件如下所示。

```
SSLCertificateFile /data/Web/KT-DMS/apache2/conf/ssl.crt/server.crt
SSLCertificateKeyFile /data/Web/KT-DMS/apache2/conf/ssl.key/server.key
#SSLCACertificateFile /data/Web/KT-DMS/apache2/conf/ssl.crt/ca-bundle.crt
```

把这些文件重命名（用 `mv` 命令），以 `.orig` 结尾，避免毁坏这些原始文件。接着，把 `dms.example.key` 复制为 `/data/Web/KT-DMS/apache2/conf/ssl.key/server.key`，把 `dms.example.cert` 复

制为/data/Web/KT-DMS/apache2/conf/ssl.crt/server.crt。我们必须了解两点，首先，需要把认证授权添加到 ca-bundle.crt 文件，然后，需要把“SSLCACertificateFile/data/Web/KT-DMS/apache2/conf/ssl.crt/ca-bundle.crt”这一行注释掉。

现在我们拥有一个包含认证授权的文件 cacert.pem，然后把它追加到 ca-bundle.crt 文件的末尾，如下所示。

```
$ sudo cat ca-bundle.crt >> /data/Web/KT-DMS/apache2/conf/ssl.crt/ca-bundle.crt
```

接着输入“sudo vi /data/Web/KT-DMS/apache2/conf/ssl.conf”命令，并从“#SSLCACertificateFile/data/Web/KT-DMS/apache2/conf/ssl.crt/ca-bundle.crt”这一行中删掉#号。

在文件/data/Web/KT-DMS/apache2/conf/ssl.conf 中，需要使用以下命令设置 ServerName 和 ServerAdmin。

```
$ sudo vi /data/Web/KT-DMS/apache2/conf/ssl.conf
ServerName dms.example.com:443
ServerAdmin Webadmin@example.com
```

最后，重启服务，使这些更改生效。

```
$ sudo /etc/init.d/kt-dms restart
```

一旦完成上述操作，重启 KnowledgeTree 服务，并测试连接 HTTPS 服务 https://dms.example.com。

12.4.8 资源

要想获得关于 KnowledgeTree 的内容，请访问以下网页。

- <http://docs.knowledgetree.com/manuals/ag/>（管理手册）
- <http://docs.knowledgetree.com/manuals/ug/>（用户指南）
- http://wiki.knowledgetree.com/Main_Page

12.5 打印服务

在 Red Hat 和 Ubuntu 上架设打印服务是非常简单的事情。这两个发行版本使用同一个打印服务器软件来管理打印服务。下面介绍如何建立 CUPS 打印服务器。CUPS 打印服务器实际上是 Linux 系统上的标准打印服务器，它在 Red Hat 主机和 Ubuntu 主机上有一致的图形用户界面。

CUPS 可以和 Samba 很好地集成到一起，从而使打印机可以被 Samba 域使用。所以，把主机作为一台打印服务器为域中的客户端提供打印服务是一件简单的事情。

12.5.1 CUPS

CUPS 在 Red Hat 上是默认安装的，同样也可以在 Ubuntu 系统上安装它。一旦安装 CUPS，

就可以利用文本文件或者 CUPS 提供的 Web 界面配置它。

选择该 CUPS 打印机工具，因为 Ubuntu 和 Red Hat 两系统都提供它。接下来，我们将首先介绍如何配置通过 USB 接口连接到主机的新打印机，然后介绍如何与网络中的其他主机共享打印机。为了说明这点，我们把 Samba 服务器所共享的打印机连接到域中的 Windows 客户端。

安装打印机管理工具和配置 CUPS

可以通过命令行或手动编辑 CUPS 配置文件（不推荐这么做）的方式配置打印机。不过，使用 Red Hat 和 Ubuntu 两者都提供的打印机配置工具进行配置则要简单得多。下面需要为 Red Hat 安装 system-config-printers 软件包，为 Ubuntu 安装 system-config-printer-common 和 system-config-printer-gnome 软件包。

接上打印机时，Linux 内核将发送一个事件通知 Udev 和 HAL（Hardware Abstraction Level manager）。Udev 是用户态的设备管理程序，而 HAL 为桌面程序提供一个统一接口。这意味着有了清晰一致的规则集合，像字处理器这样的应用程序就可以知道如何使用打印设备，日历程序可以也能无障碍访问 PDA（掌上电脑）。

Linux 上的设备管理

主机接上一个新设备后会发生什么？系统怎么知道如何处理它？当内核识别了新的设备，它将把设备信息写入文件系统中的 /sys 目录下，然后给消息总线发送事件通知。在消息总线上监听的有 Udev 的后台进程 udevd 和 HAL（硬件抽象层：Hardware Abstraction Level manager）管理程序。

Udev 是用户空间设备管理器，它能被一组规则文件配置。Udev 有一个数据库，用来记录系统中的所有设备。Udev 会读取其规则列表，查找关于如何处理新设备的所有特殊指令。如果 Udev 找到一条规则，便执行该规则，否则只是在 /dev 下根据新设备的类型创建一个设备文件。一般磁盘会被分配一个 /dev/hda 或者 /dev/sdb 这样的设备文件，网卡会被分配 eth1 或者 eth2 这样的设备文件。当设备被删除时，会发生同样的事情。内核将发送一个事件通知给 udevd 后台进程，并且根据是否有符合该设备的相应的规则来执行某些动作。

HAL 是为使用底层硬件而必须了解硬件信息而设计的、用来解放桌面应用程序的软件。HAL 将给桌面应用程序提供有关打印机、PDA、USB 存储设备等的有效信息，所以这些桌面应用程序可以通过给定的 API 利用这些设备。这样做有两个好处：设备访问变得一致，应用程序设计也变得更简单。

基本上，用户不需要对 Udev 和 HAL 了解太多的内容，因为它们都处于底层。然而，如果感兴趣可以到以下网址阅读更多内容。

- <http://www.freedesktop.org>
- <http://www.redhat.com/magazine/002dec04/features/udev/>

试着把一个 Epson USB 打印机插入 Ubuntu 主机看看会发生什么。为了确定一切都工作良好，首先执行 tail 命令查看系统日志，确定该设备已经被内核所识别。

```
$ sudo tail /var/log/syslog
Feb 18 21:27:16 au-mel-ubuntu-1 kernel: [ 2561.902999] usb1p0: USB Bidirectional ➡
printer dev 10 if 0 alt 0 proto 2 vid 0x04B8 pid 0x0005
Feb 18 21:27:16 au-mel-ubuntu-1 kernel: [ 2561.903329] usbcore: ➡
registered new interface driver usb1p
```


这里可以看到，主机已识别了 USB 打印机，并用 `usbip` 注册了该 USB 打印机。通过输入 `ls /dev/usbip` 命令，可以检查该设备文件是否存在。现在可以满意了，因为设备已经被识别，并且即将被添加到 CUPS 打印服务的控制之下。

如今，配置打印机设备是相当简单的事。首先，从桌面菜单中选择菜单“系统”>“管理”>“打印”，这将打开打印机配置工具。该工具如图 12-47 所示。

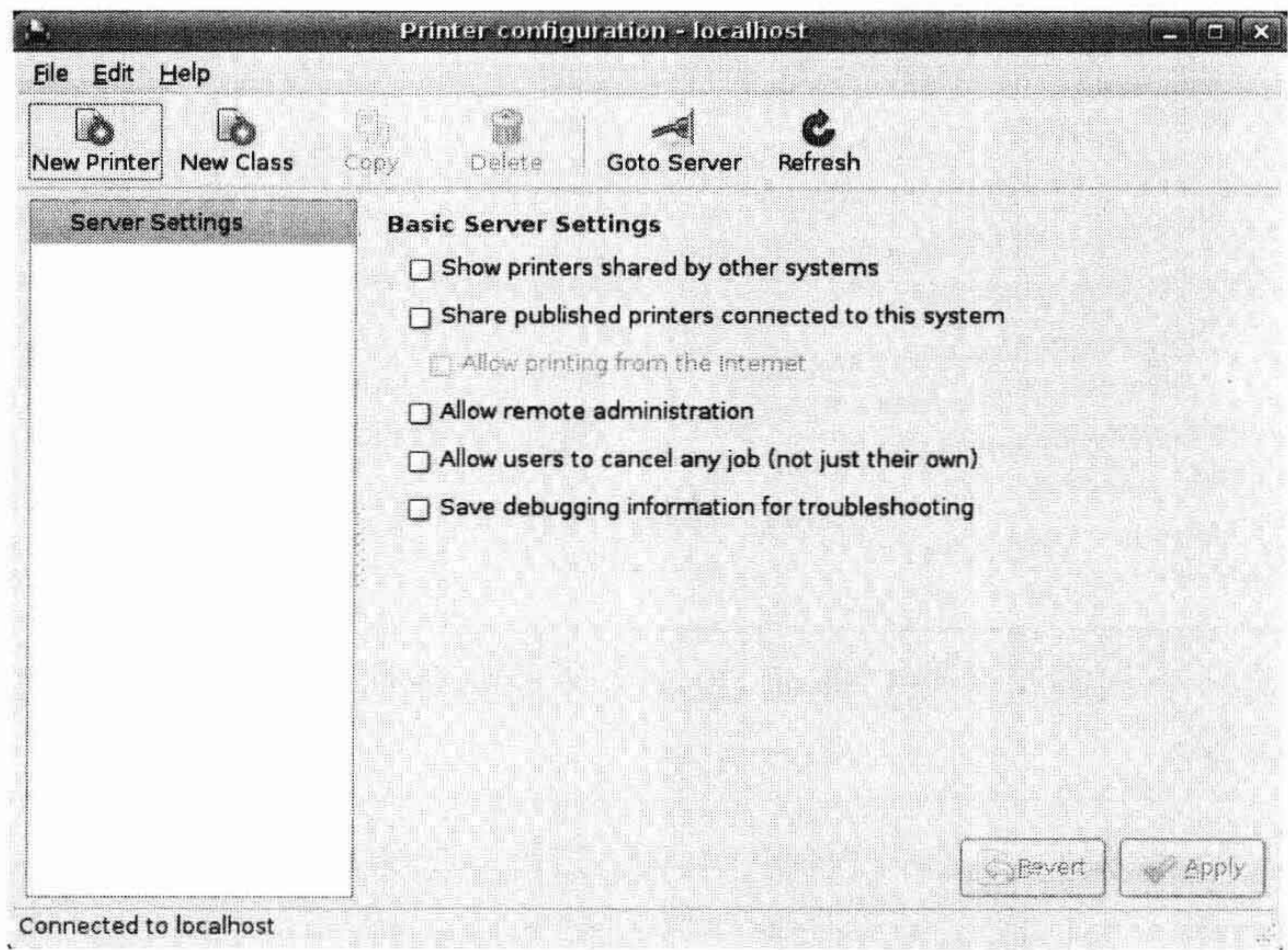


图 12-47 打印机配置工具

如上图所示，目前还没有设置任何打印机。单击“New Printer”图标添加打印机。这会弹出如图 12-48 所示的窗口。

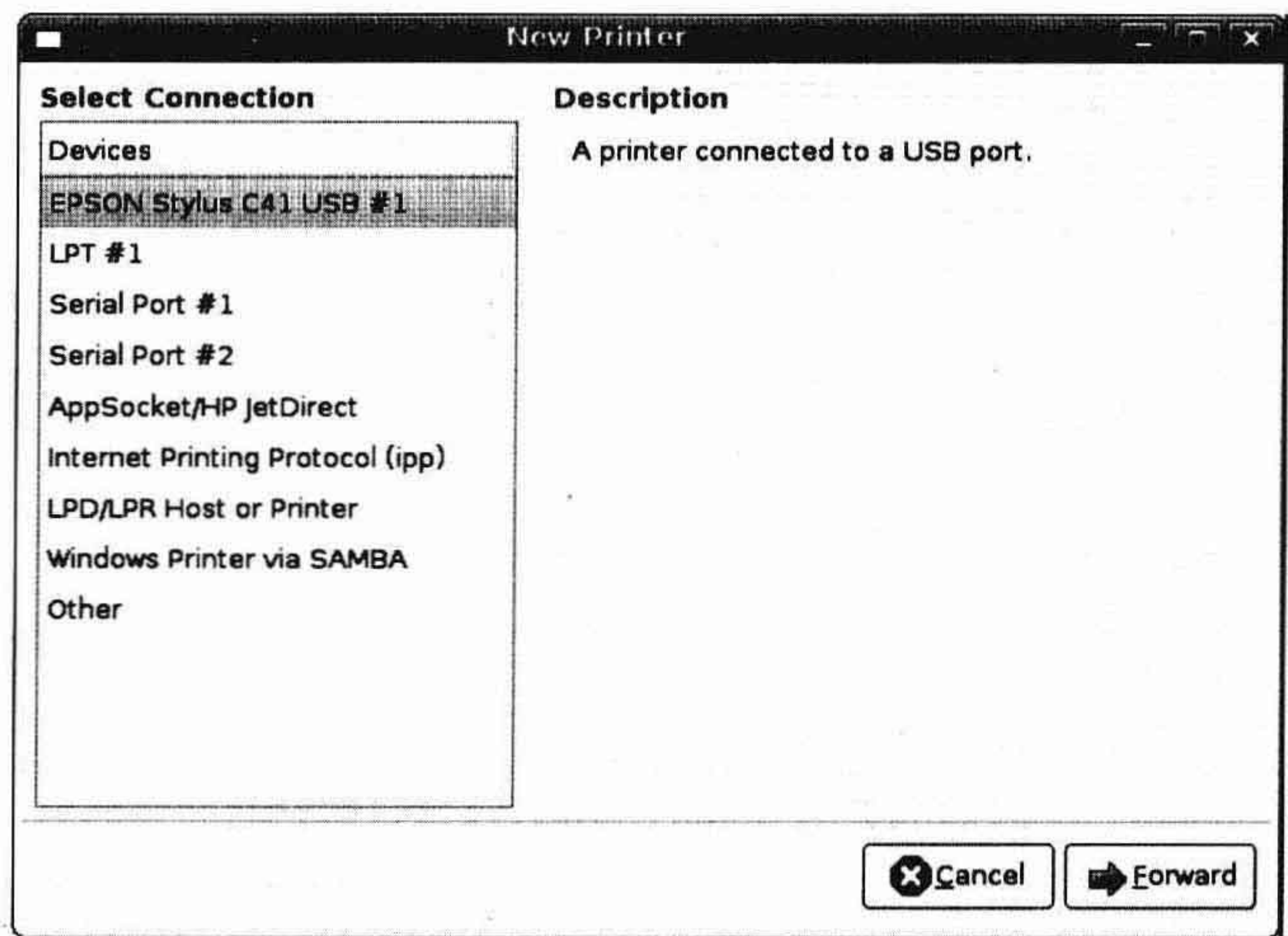


图 12-48 添加打印机

可以看到 Ubuntu 主机已经找到新近连接的打印机。选中该打印机，并单击“Forward”按钮，弹出如图 12-49 所示的窗口。

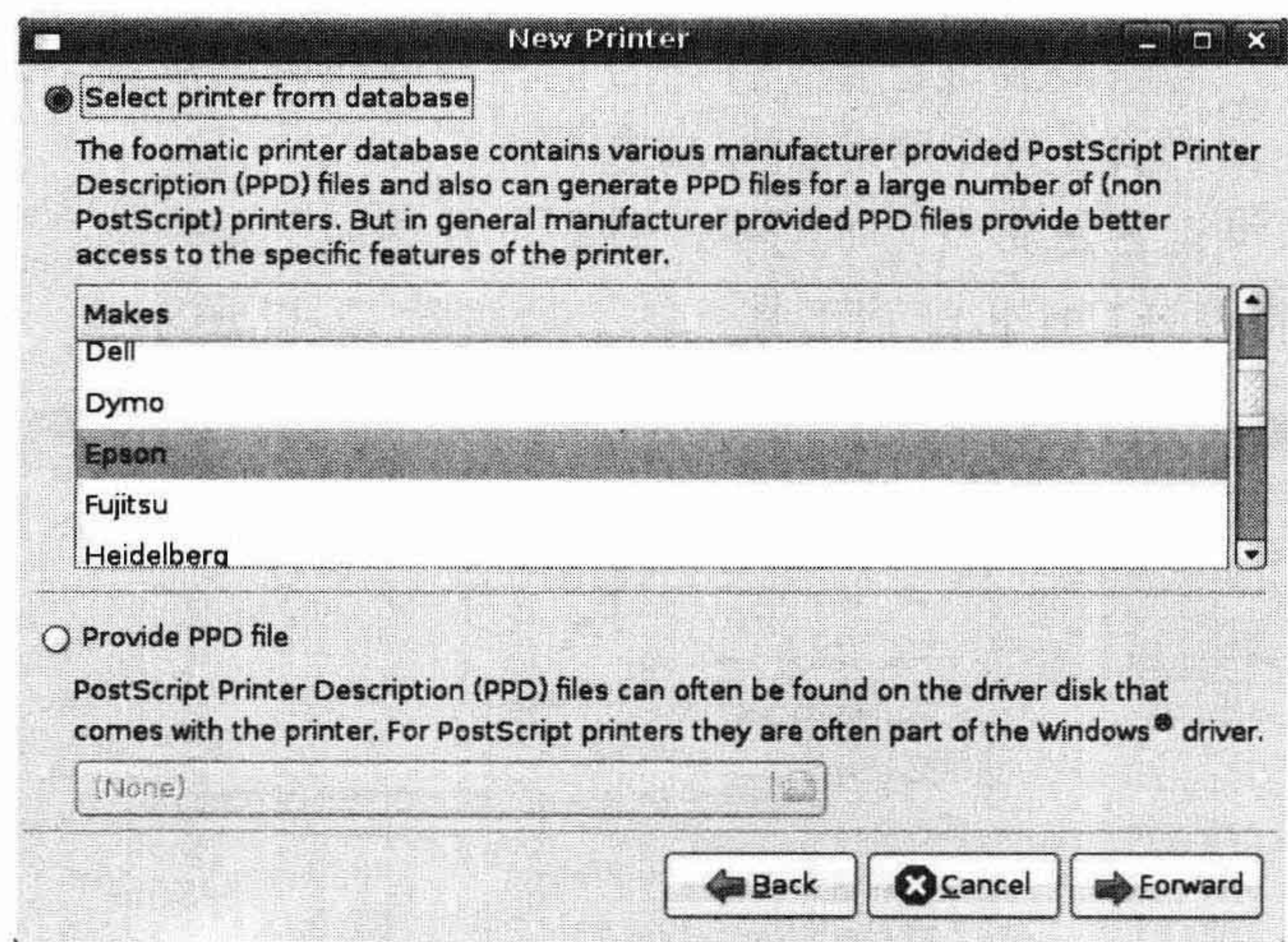


图 12-49 选择一个打印设备

■注：Foomatic 是一个捆绑了各种厂商的 PPD 文件的软件包。这些文件为 CUPS 提供了同打印机交互的正确方式。

接下来把刚接上系统的 Epson 打印机用作打印机驱动程序。单击“Forward”按钮弹出一个窗口。从该窗口中，为打印机型号选择一个匹配的驱动程序（见图 12-50）。高亮的打印机驱动程序与我们的打印机所最推荐的。单击“Forward”按钮，看到如图 12-51 所示的窗口。在这里，为该打印机设置名称、描述和位置等详情，然后单击“Apply”按钮。

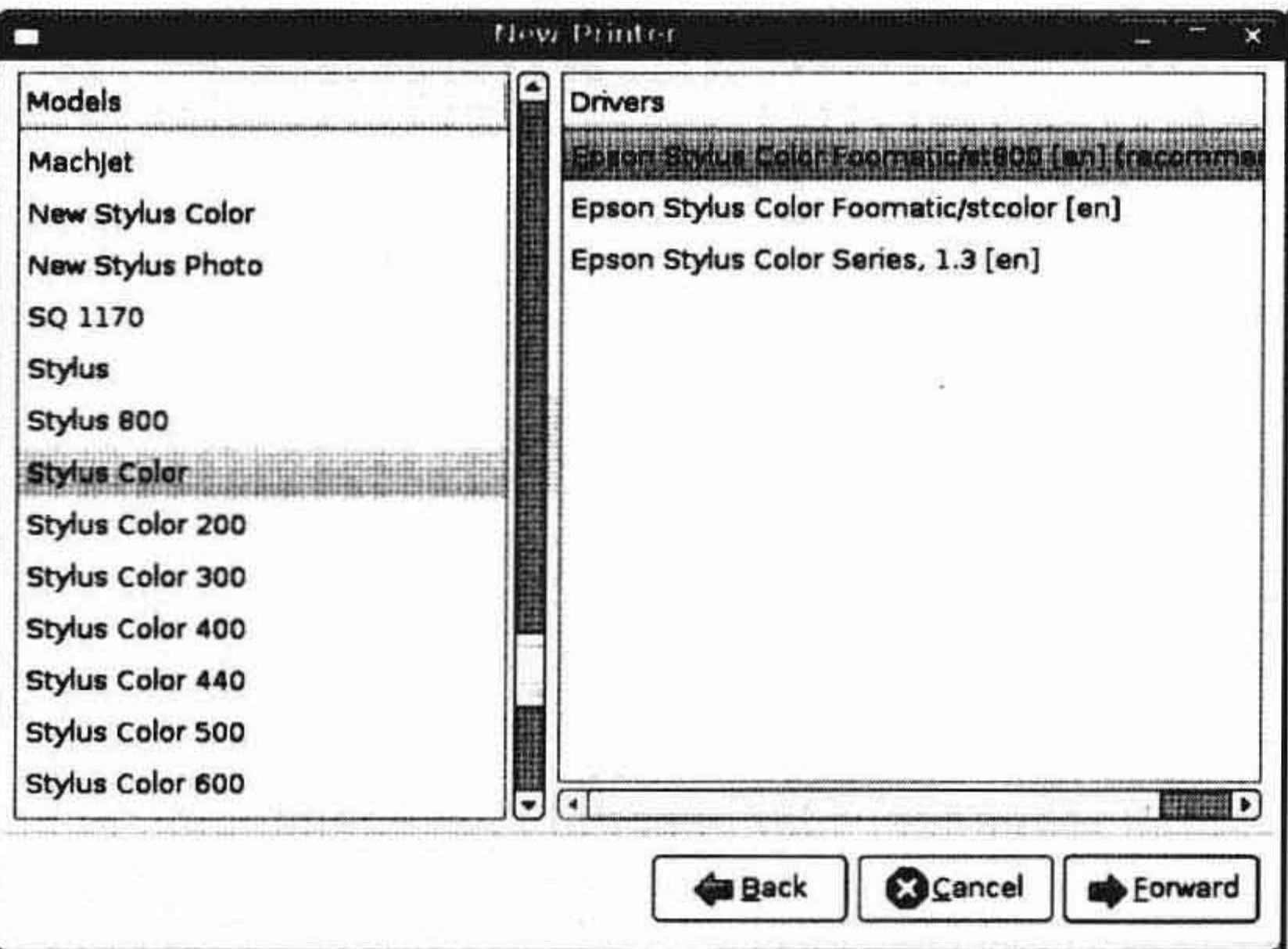


图 12-50 为打印机选择驱动器

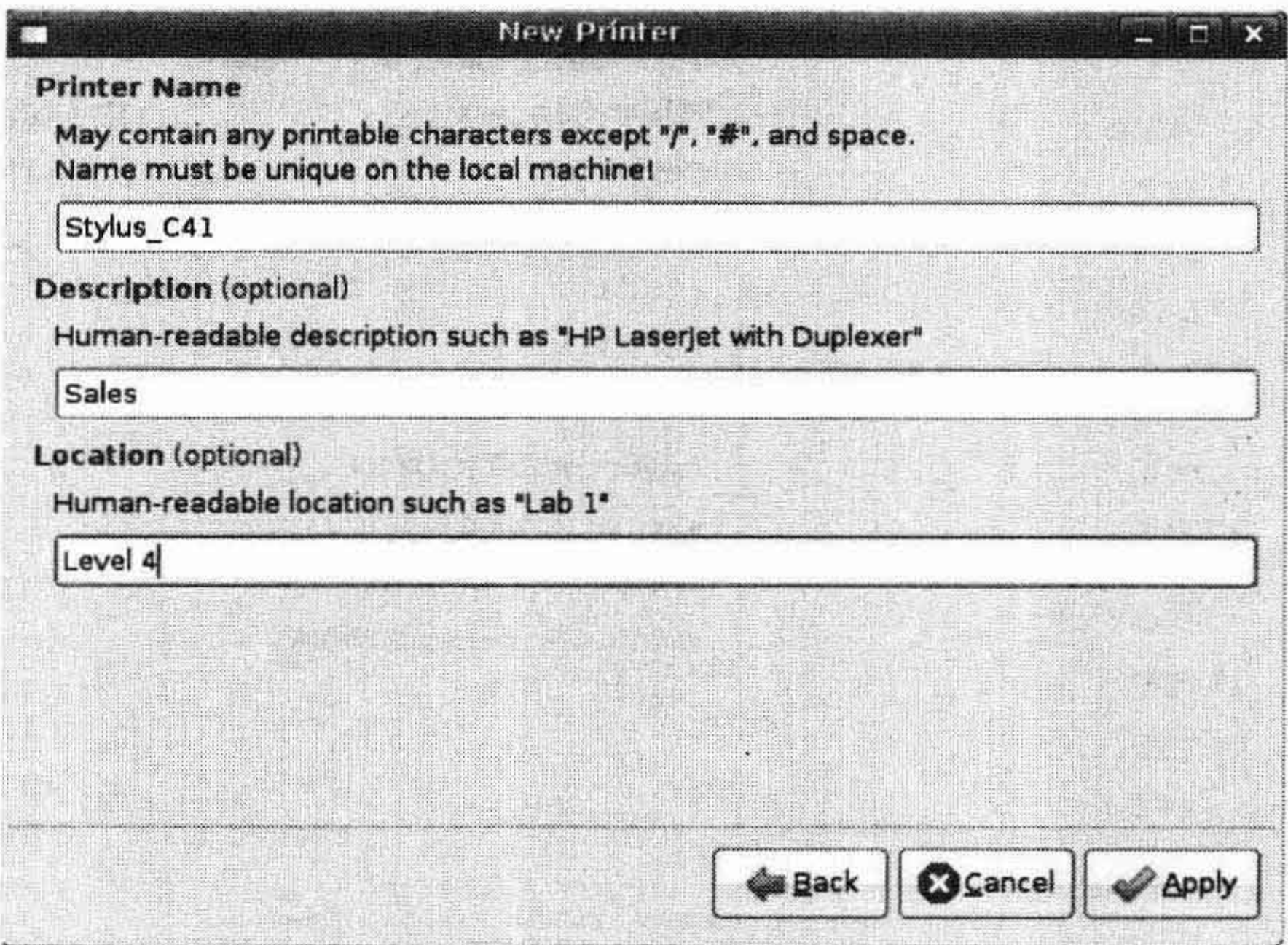


图 12-51 打印机名称等详情

在图 12-52 中，可以看到该打印机被添加到“Local Printers”区域下。单击“Local Printers”区域下的“Stylus_C41”，可以看到打印机的详细信息，如图 12-53 所示。如果需要，可以编辑这些详细信息。

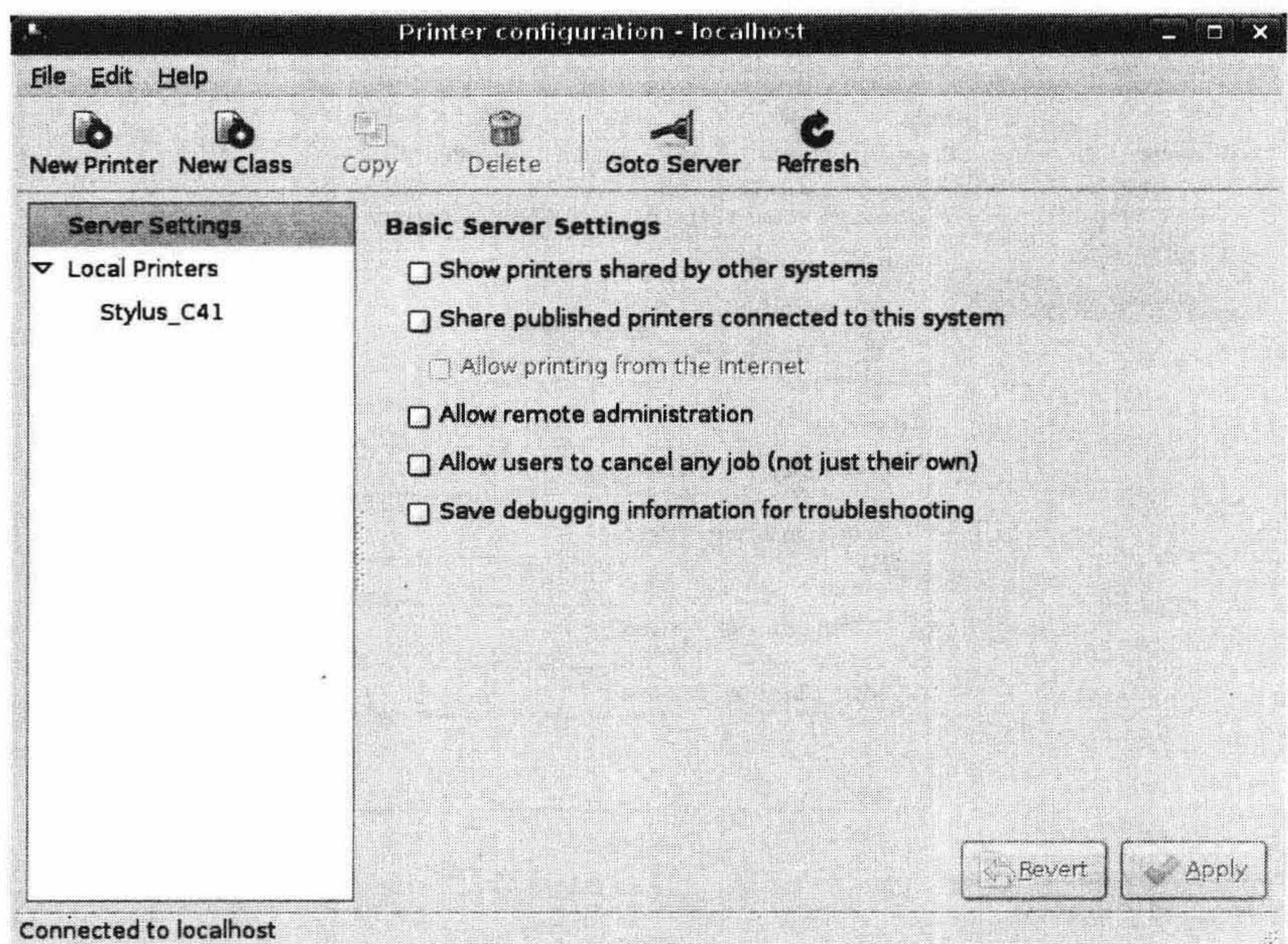


图 12-52 打印机已经被添加了

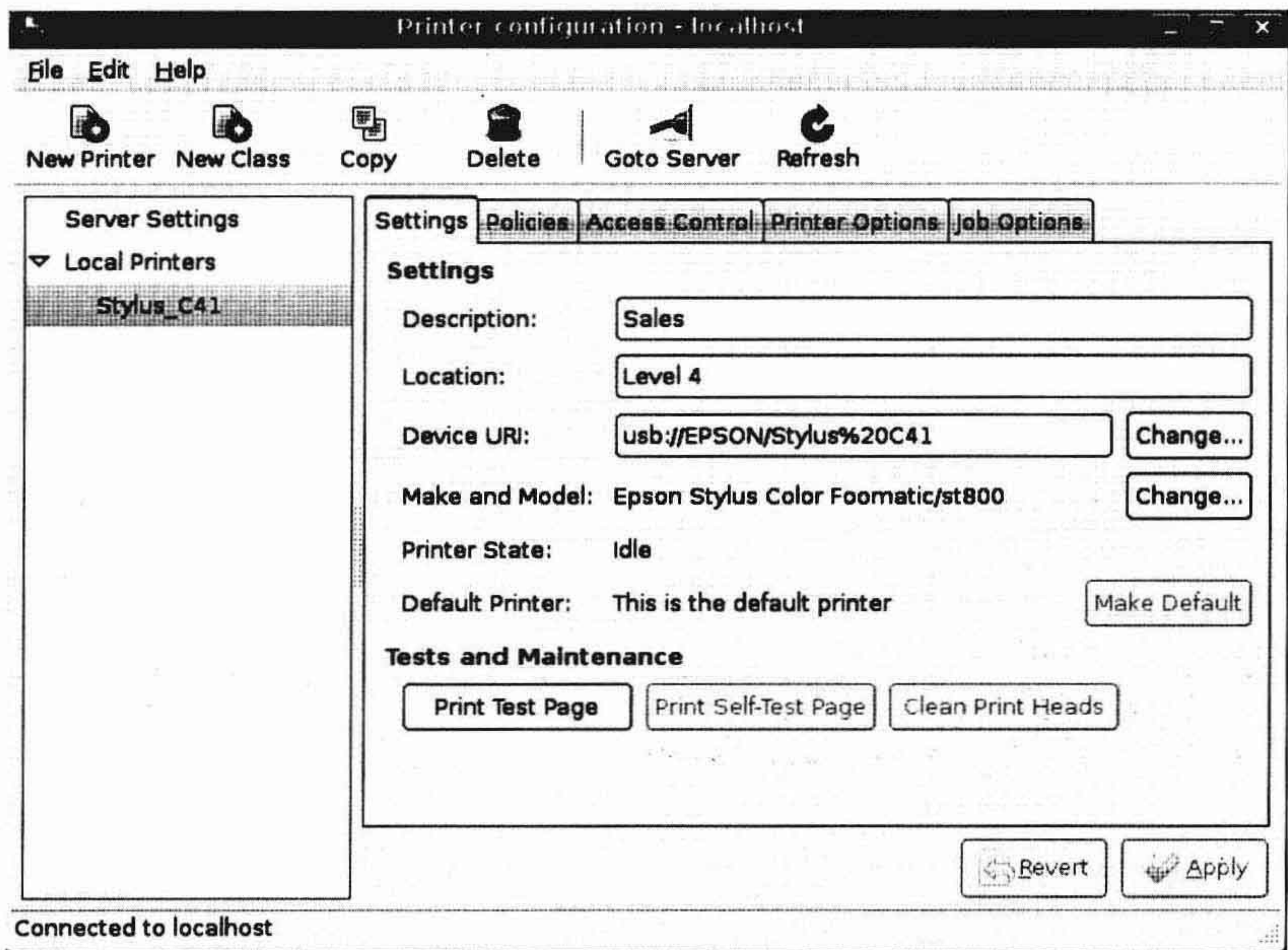


图 12-53 所安装的打印机的属性

可以看到面板上的描述和位置正是刚刚给打印机设备所设置的。“device URI”是由接口类型和打印机的位置组合而成的。在这里，可以用“Print Test Page”按钮测试打印。

如图 12-54 所示，通过单击“Polilcies”标签页，可以设置打印机状态为启用或者不启用，还可以设置其他一些选项。

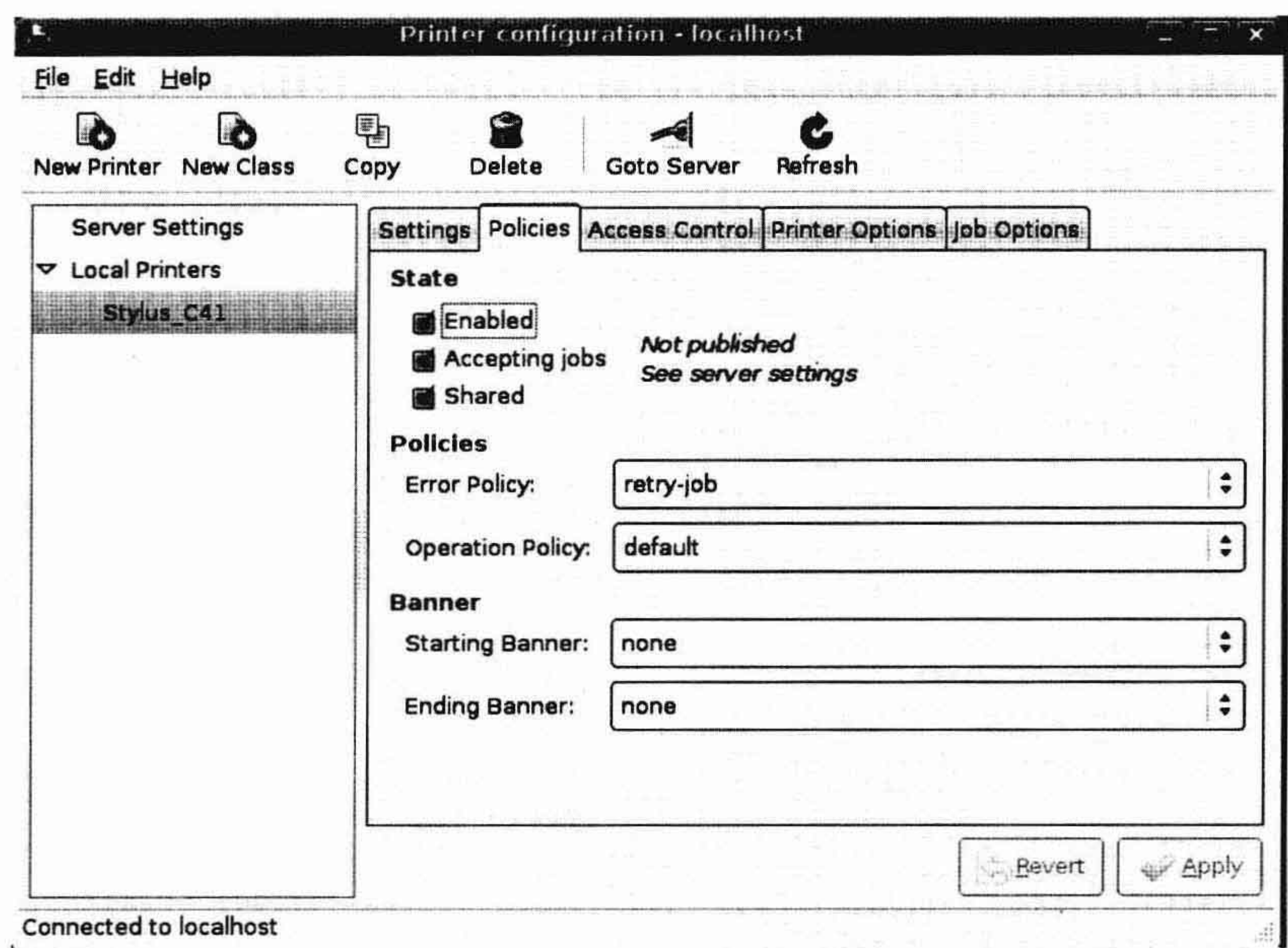


图 12-54 打印机策略

在图 12-55 中，可以在“Access Control”标签页中为主机设置一些安全措施，拒绝某些用户使用打印机。当然，所拒绝的用户在主机上必须有个账号，否则这些设置将不会生效。

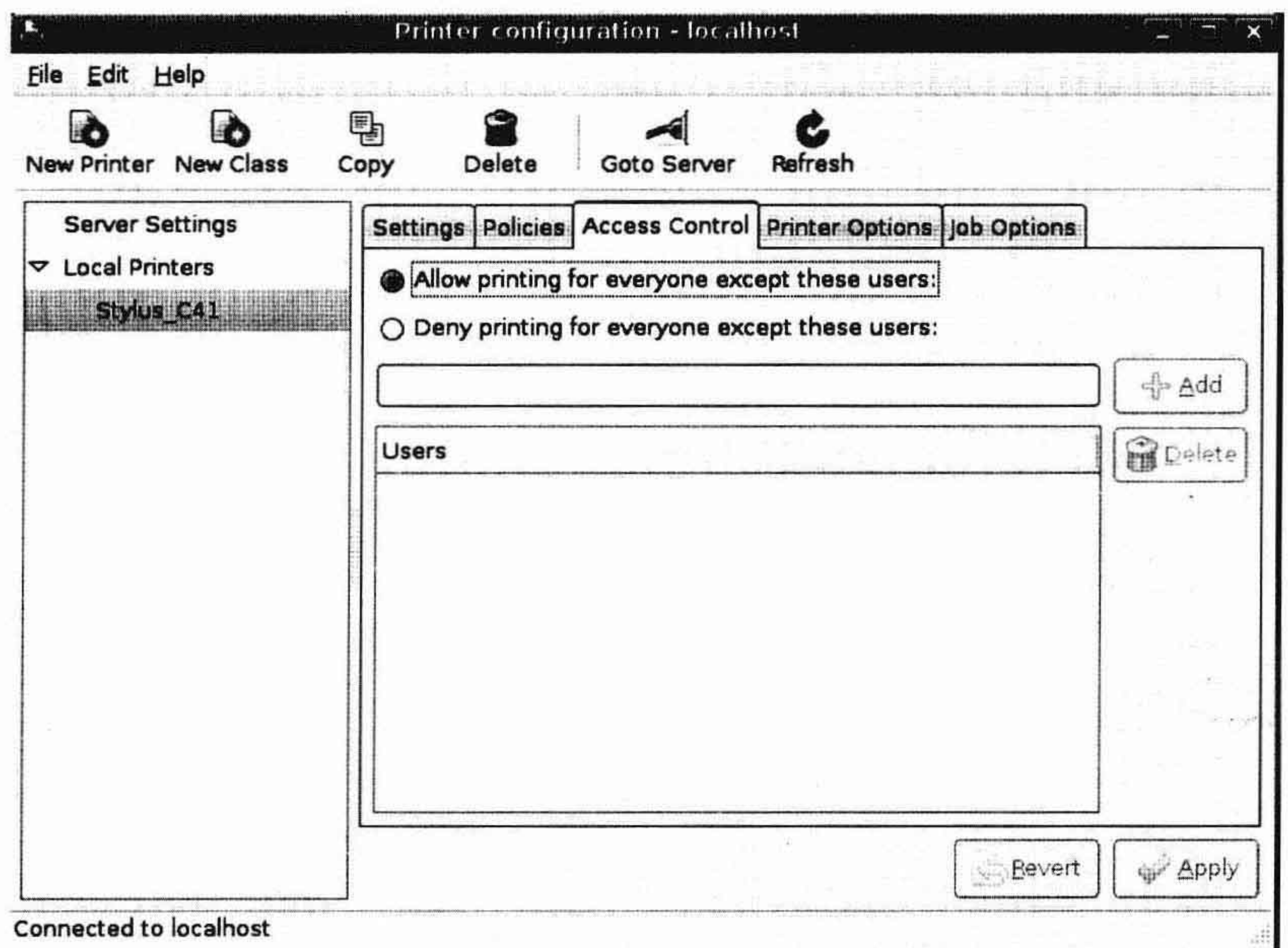


图 12-55 打印机访问控制

在“Print Options”标签页，可以设置打印机的页面大小和默认分辨率（见图 12-56）。

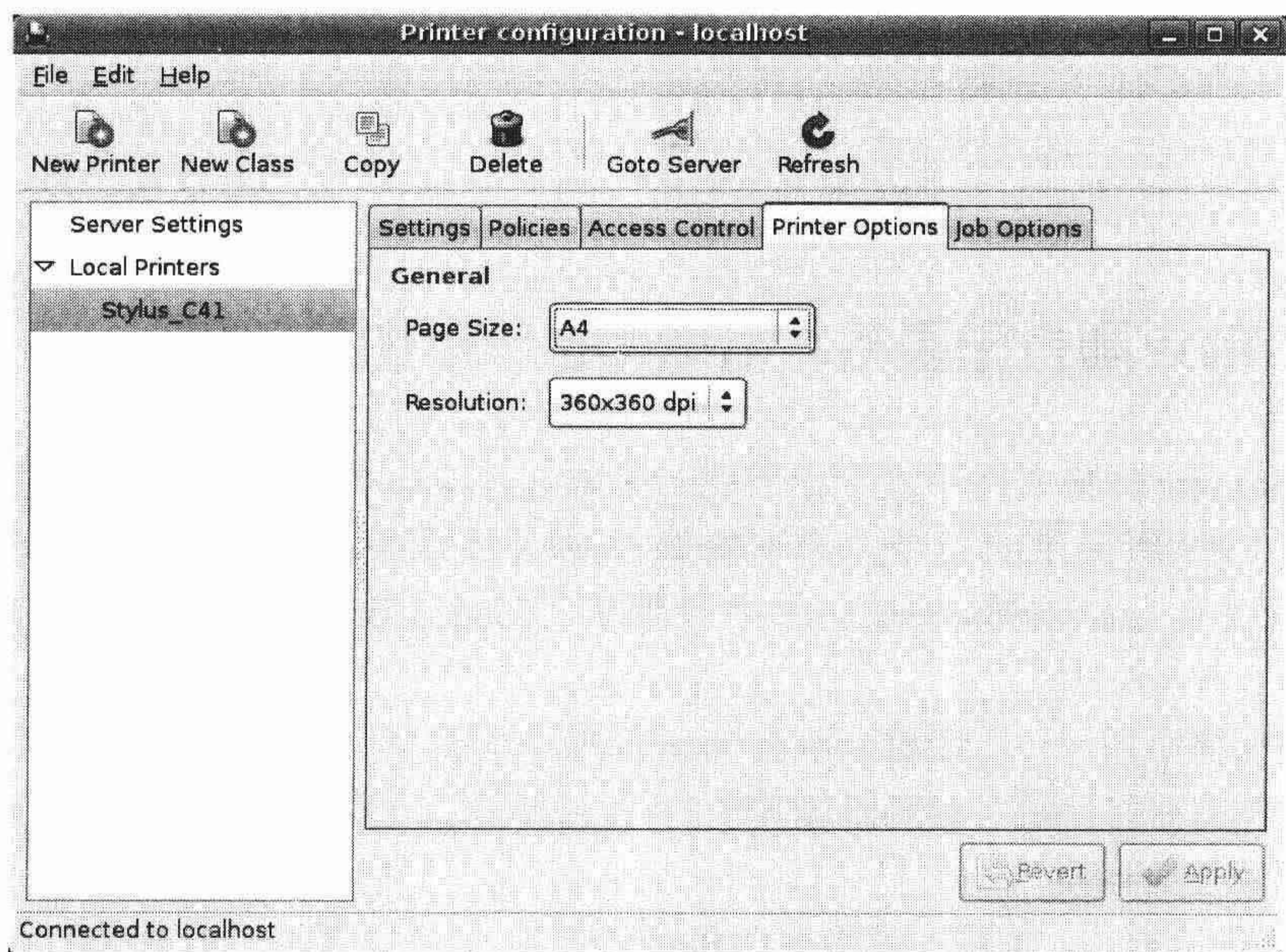


图 12-56 打印机页面默认值

打印作业选项可以在“Job Options”标签页找到。如图 12-57 所示，它允许设置多种作业选项。

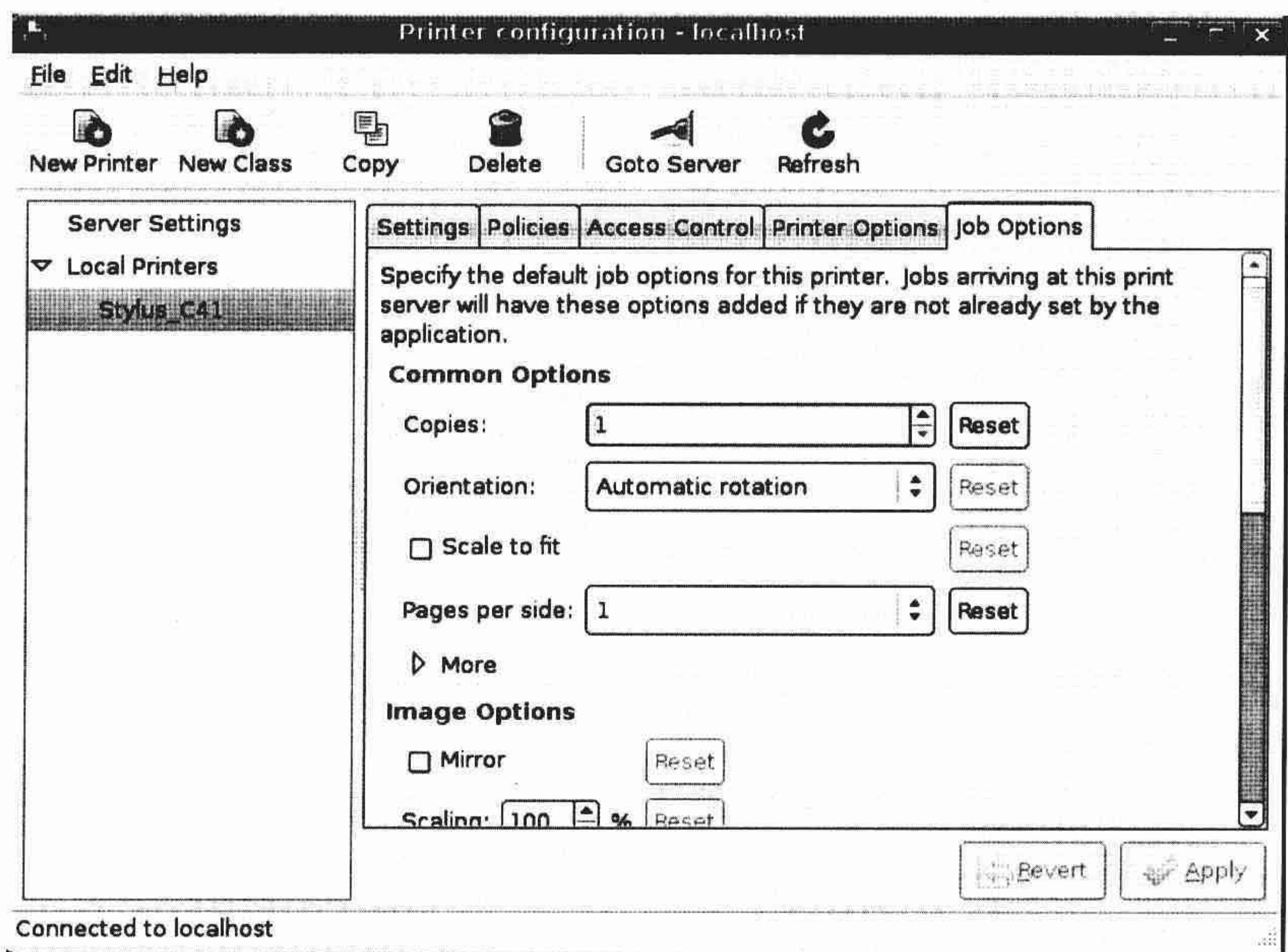


图 12-57 打印机作业选项

现在，选择菜单“File”>“Quit”退出打印机设置工具。重启 CUPS 打印机服务器，确保所做的配置已经被读取。

现在，根据主机所使用的 Linux 版本，用熟悉的 `service` 或 `invoke-rc.d` 命令来重启打印机服务器。

```
$ sudo /sbin/service cups restart
$ sudo /usr/sbin/invoke-rc.d cupsys restart
```

12.5.2 Samba 和打印服务：给桌面系统添加打印机

可以在 Samba 配置文件中看到 `[printer]` 服务和 Samba 包含的打印服务指令，而这些指令类似于 CUPS 所提供的指令。为了确保 Samba 成功载入所有新添的打印机，必须确保已经在配置文件 `/etc/samba/smb.conf` 中设置了 `load printers = yes`，并重启 Samba 服务器。

现在介绍如何添加一台打印机到不在 Samba 域中的共享主机。登录到 EXAMPLE 域，为该桌面系统添加那台打印机，这和添加其他的网络打印机一样。首先，打开“打印机和传真”菜单并单击“添加打印机”。

在弹出的对话框中，选择“网络打印机或连接到其他计算机的打印机”（见图 12-58）。下一步，选择“浏览打印机”，如图 12-59 所示。

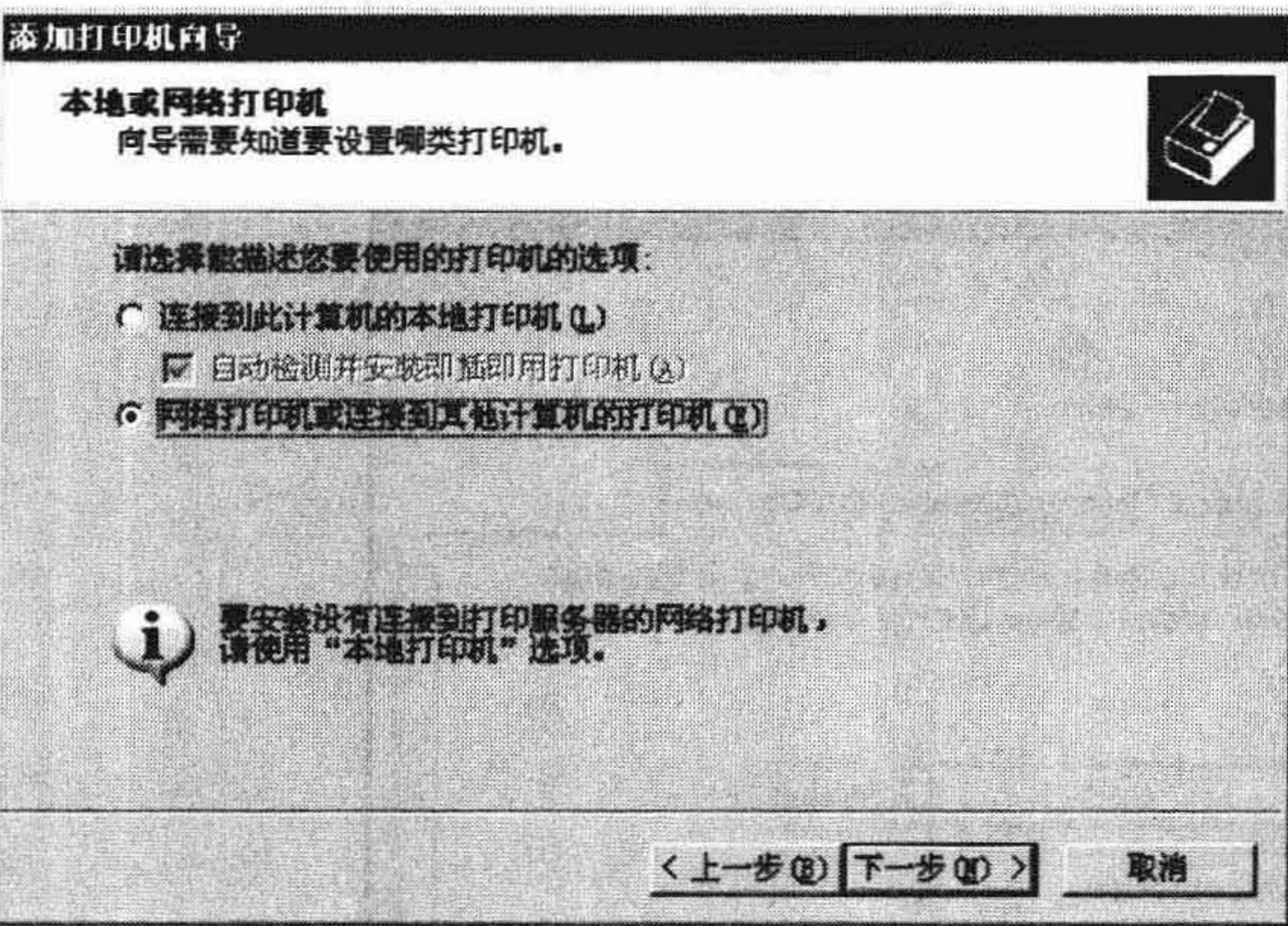


图 12-58 添加网络打印机

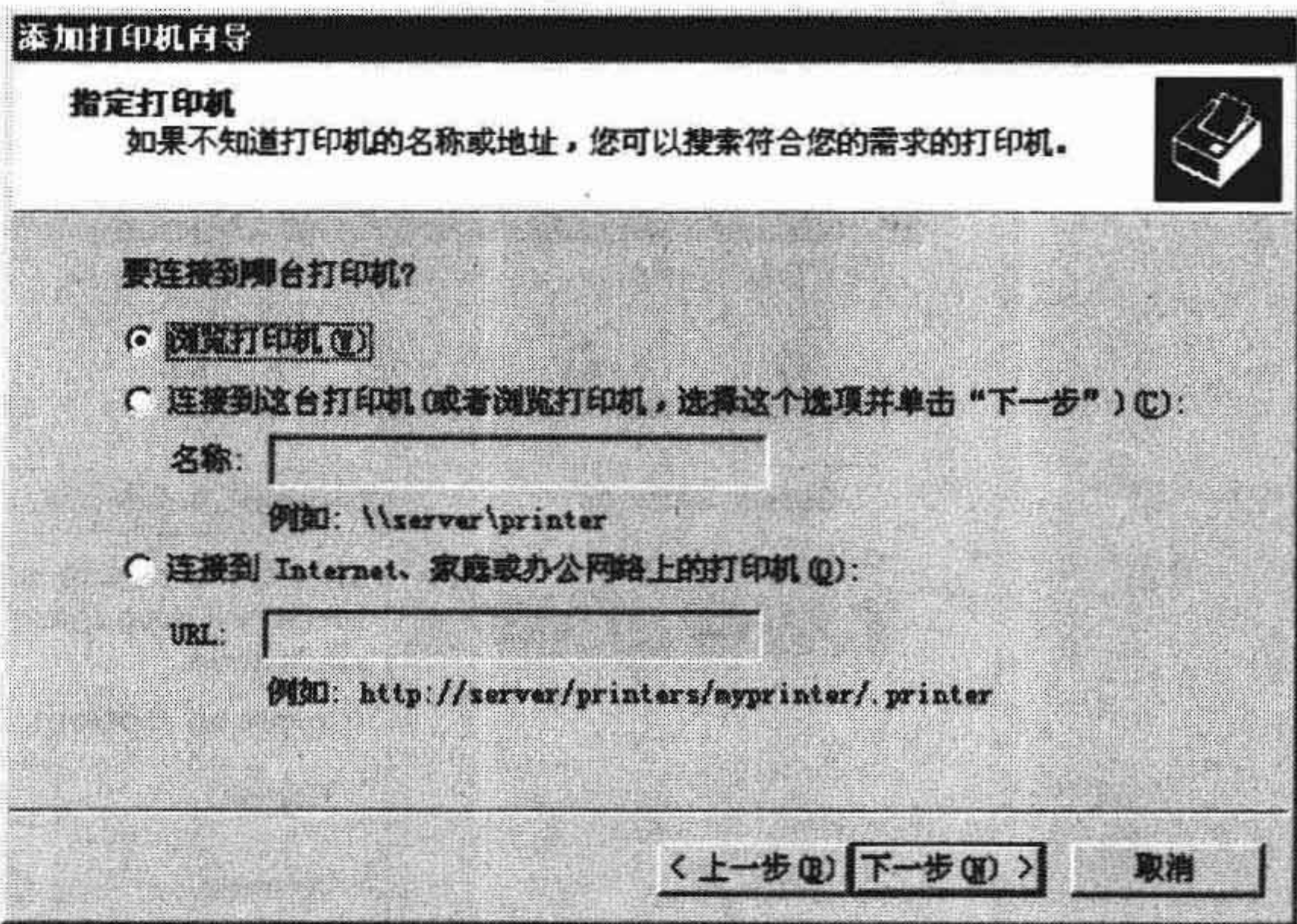


图 12-59 浏览打印机

图 12-60 显示该 Epson 打印机已连接到 AU-FILESERVER-1。我们希望用 AU-FILESERVER-1 作为网络打印机。

在添加打印机这一步，如果还没有安装打印机驱动器，则需要选择一个打印机驱动器。如果没有可用的驱动器，则可以一直选择“Generic Text printer”驱动器，或者从打印机的官方网站获取驱动器。

现在，打印机已连上主机并做好了打印准备，如图 12-61 所示。
到此，我们安装了一台通过 Samba 服务器共享的打印机。

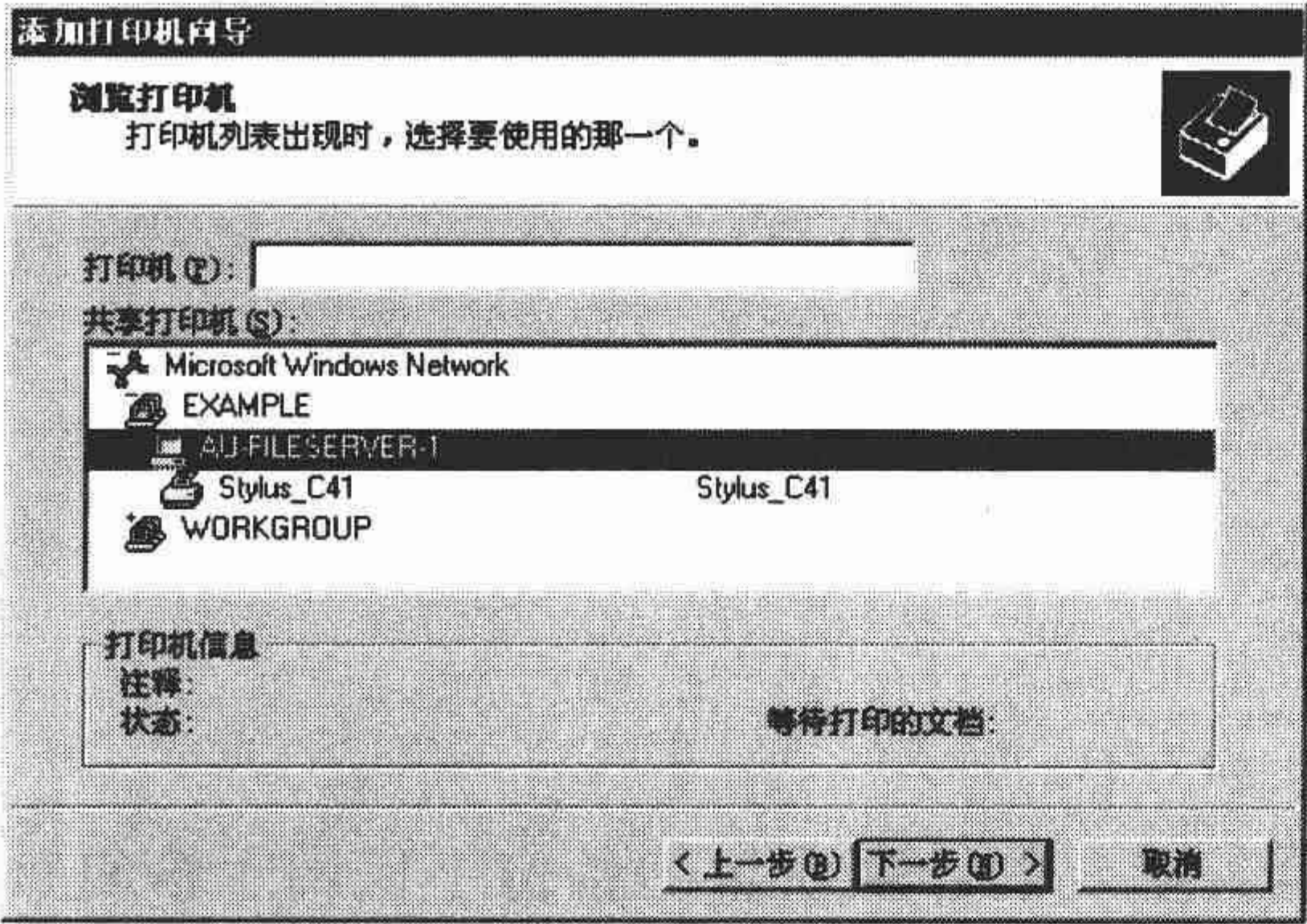


图 12-60 选择打印机

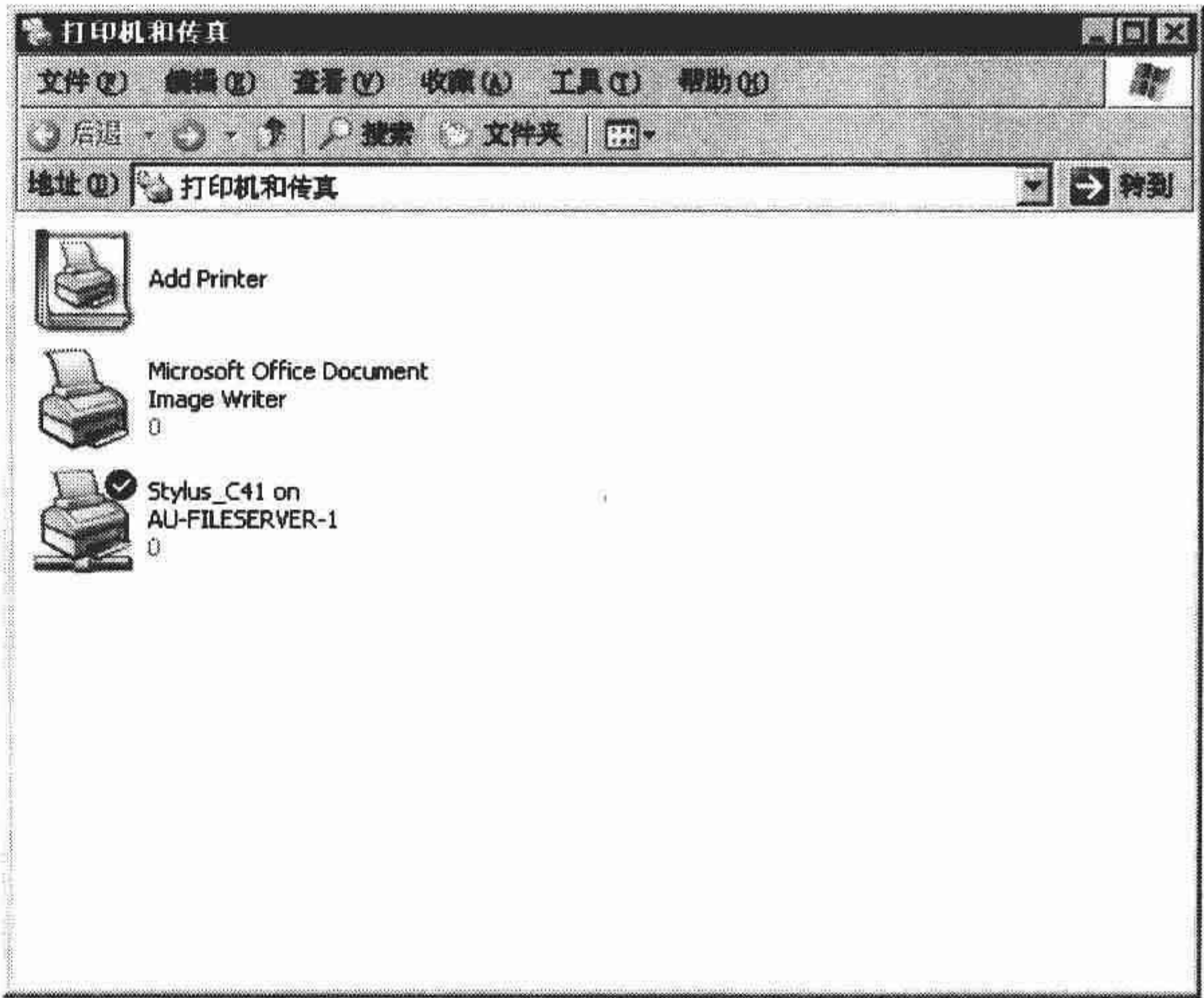


图 12-61 安装好的打印机已准备工作

12.6 小 结

本章探讨了 Linux 提供的文件和打印共享服务，介绍了传统的文件共享方法以及 KnowledgeTree 文档管理系统。传统的文件共享通过 Samba 服务器或 NFS 服务器实现，具体选择哪种方法依赖于请求服务的客户端。KnowledgeTree 文档管理系统是传统的文件共享方法之外的一个非常好的选择。相比文件共享，DMS 具有许多优势，因为它提供了文档细粒度的访问控制和版本控制，并能实现企业的工作流。最后，介绍了如何建立 CUPS 打印服务以及如何通过 Samba 服务器共享它。

通过本章学习，读者应该具备以下能力。

- 配置 Samba 服务器来控制一个域。
- 配置 NFS 服务器在 Linux 主机之间挂载文件系统。
- 安装和配置 KnowledgeTree DMS。
- 在 DMS 中检出和检入文档。
- 配置 CUPS 打印机服务器并把它们连接到同一域的主机上。

下一章将介绍备份和恢复。

第 13 章 备份和恢复

...

备份和恢复数据的能力对任何企业而言都是至关重要的。高质量的备份不仅用于恢复被删除或被覆盖的数据，还可用于政府法律要求的案例（例如，与保存税收记录和消费记录相关的案例）。

我们以讨论灾难恢复计划（DRP）和业务持续管理（BCM）开始本章，使读者对这些概念有一个基本的认识；然后向读者展示如何安全地复制远程主机上的数据，无论主机是位于本地网络还是远在世界的另一端；接着介绍备份服务器 Bacula，展示如何使用它保存与恢复文件；另外还将涉及如何用 Bacula 备份与恢复数据库的相关内容；最后讨论 Bat 控制台，它是 Bacula 的一种图形用户接口。

本章结束时，读者应该具备以下能力。

- 了解 DRP 和 BCM 的必备条件。
- 使用 rsync 命令把数据从一个主机安全地复制到另一个主机，并利用脚本自动化该过程。
- 安装和配置 Bacula 备份服务器。
- 使用 Bacula 管理备份和创建备份任务。
- 使用 Bacula 恢复主机上的文件。
- 配置和使用 Bat 控制台。

接下来开始总体讨论 DRP。

13.1 灾难恢复计划

我们当然希望自己的业务不会发生任何灾难性故障，但是为了以防万一，最好对任何情况都有所准备。灾难主要有两种类别：人为灾难和自然灾难。一台电子邮件服务器一整天工作失灵，导致极其重要的、时效性敏感的商务邮件丢失，这就是一种与人为错误或机器故障有关的人为灾难，这种灾难可以通过相关的恢复过程恢复。另外，一场摧毁了办公室的地震是一种自然灾难，这种灾难需要完全不同的恢复应对措施。根据这两种灾难事件发生的可能性，可以对它们制定应对计划。

灾难恢复计划 (DRP) 是指关于识别、管理和降低灾难风险的一切事情。它是被称作业务持续管理 (BCM) 的拱型流程一部分。业务持续管理也可解释为在面对未知灾难的时候确保业务至少按照预定的最低水平持续运作。BCM 涉及企业的众多方面, 它需要详细列出企业承诺的特定服务的恢复时间表。

下面是在制定 BCM 和 DRP 策略时需要考虑的一些问题。

- 能够预测业务最有可能面临的干扰吗? 从这些干扰中恢复的必要步骤有哪些? 预期的恢复时间表是什么?
- 降低灾难风险并从每一个可能的灾难事件中恢复的代价是多少?
- 是否需要一台主机托管, 以便在灾难发生的时候把业务移到那里?
- 在任何可能的危急关头, 是否需要租用额外设备, 例如发电机?
- 当业务发生灾难性事件时, 需要同哪些组织和人员进行接触和沟通? 应该如何告知外界这些灾难性事件?
- 在大规模灾难性事件发生的情况下, 决定业务的持续性是否可以实现的因素是什么? 如果关键性基础设施或者商业财产受到损害, 灾难损失将会迅速增加。

制定 BCM 和 DRP 计划是一个复杂的过程。企业制定的 BCM 计划应该包括风险分析结果、业务影响分析和由业务主管部门批准的危机管理调查。即使是小型业务也可以从半正式的 BCM 计划中获益, 虽然制定完整的 BCM 计划所需的资源可能并不具备。关于 BCM 和 DRP 的更多信息, 我们推荐以下资源。

- Ready Business(U.S. Homeland Security):<http://www.ready.gov/business/index.html>
- Australian National Security ,Business Continuity Planning: http://www.ag.gov.au/agd/www/nationalsecurity.nsf/Page/Information_For_BusinessBusiness_Continuity
- Wikipedia, Business Continuity Planning: http://en.wikipedia.org/wiki/Business_continuity_planning
- Business Continuity and Disaster Recovery Checklist for Small Business Owners: <http://www.continuitycentral.com/feature0501.htm>
- Disaster Recovery Using Bacula: http://www.bacula.org/en/rel-manual/Disast_Recovery_Using_Bacula.html

本章把数据的备份和恢复过程作为重点, 该过程是企业 BCM 和 DRP 计划的一部分。下一节介绍备份策略。

13.2 备份过程

在选择备份方式的时候, 需要考虑许多不同的问题。回答下面这些问题将为公司提供合适的备份策略。

- 要备份什么?
- 需要每隔多久备份一次数据?
- 要把备份数据保存多久?

- 应该把备份存储在哪里以及存储在什么介质上？

这里重要的是应了解所要备份的数据，需要知道数据每隔多久就会发生改变以及数据大小的变化，还需要了解存储介质，知道它能保存多少数据以及它的存储期限。如果没有计划，数据的大小是难以掌控的，可能无法备份过多的数据，或者没有足够的时间来备份数据。在这些情况下，我们可能发现自己正在备份不需要的数据，或者需要使用其他具有更高性能或更大备份容量的设备。

数据的保存周期也是需要慎重考虑的。出于法律或税收的目的，也许要把数据保存一段特定的时期，通常是几年。客户资料可能需要保存几年，而其他类型的数据保存时间可能要短一些。根据所要备份的内容，为了恢复数月前的数据而把数据保存在媒介上的做法可能是完全没有意义的。在这些例子中，应该考虑缩短备份周期以释放媒介的存储空间。

另外需要考虑到的一点是备份时间的安排。安排备份的可用时间窗口（Time Window）是多少？也许我们正在经营一家 24 小时商店，它留给离线备份方式的时间就非常少。那么可能不得不采用热备份（hot backup，即在某段时间某一时刻，在工作着的主机上进行备份）方式，而且可能还没有足够的时间这么做。为了安排好备份时间，也许需要考虑每日备份、每周备份或者每月备份的备份方式。

最后，需要决定所要执行的备份的类型。在大多数备份方式里，备份可以分成 3 种类型：完全备份、增量备份和差异备份。完全备份就是数据的全部备份，是占用空间最大、耗费时间最长的备份过程。增量备份就是在最近一次备份（可以是完全备份、差异备份或者增量备份）之后，对被修改的文件或目录所做的备份。增量备份小于完全备份和差异备份，而且执行起来通常更快。差异备份就是在最近一次备份之后，对所有被修改的文件做备份。差异备份比增量备份大，但是在做数据恢复的时候很有用，因为只需要完全备份和最后一次备份有差异的数据就够了（如果当前并没有执行增量备份）。在备份时间表上，应该每周做一次完全备份，同时每夜做一次增量备份或差异备份。

使用完全备份、增量备份和差异备份，恢复操作按照以下次序完成：首先恢复最后一次完全备份，接着恢复最近一次差异备份，然后恢复后续的任何增量备份。如果我们不使用差异备份，那么恢复次序就是在恢复最后一次完全备份之后，从最早到最近依次恢复每一个后续的增量备份。

■注：在备份操作中，连同需要的数据一起，我们可能还恢复了不需要的、已被删除的文件和目录。因此在继续处理之前，应该仔细检查已经恢复了哪些数据。

每个网络都有其特定的备份需求。在做备份时，有大量的硬件设备和在线存储设备可供选择。我们可以在购买硬件设备的同时购买不同厂商的商业软件，或者只购买硬件而运行开源软件执行备份。以下是一些硬件存储设备。

- 磁带：根据备份的大小选择不同的类型。
- 硬盘：有不同的转速和大小可供选择。
- 光盘：DVD/CD-ROM，用于低容量数据备份。

在线存储设备可能只适用于存储小容量数据，除非具备高速的网络带宽和便宜的网费。

13.3 网络备份

我们的示例网络很简单：一台中心主机 `headoffice.example.com`，多台位于远程分公司的主机，这些分公司的主机上有需要备份的数据。

图 13-1 展示了该示例网络（该图从第 6 章的图表演化而来）。

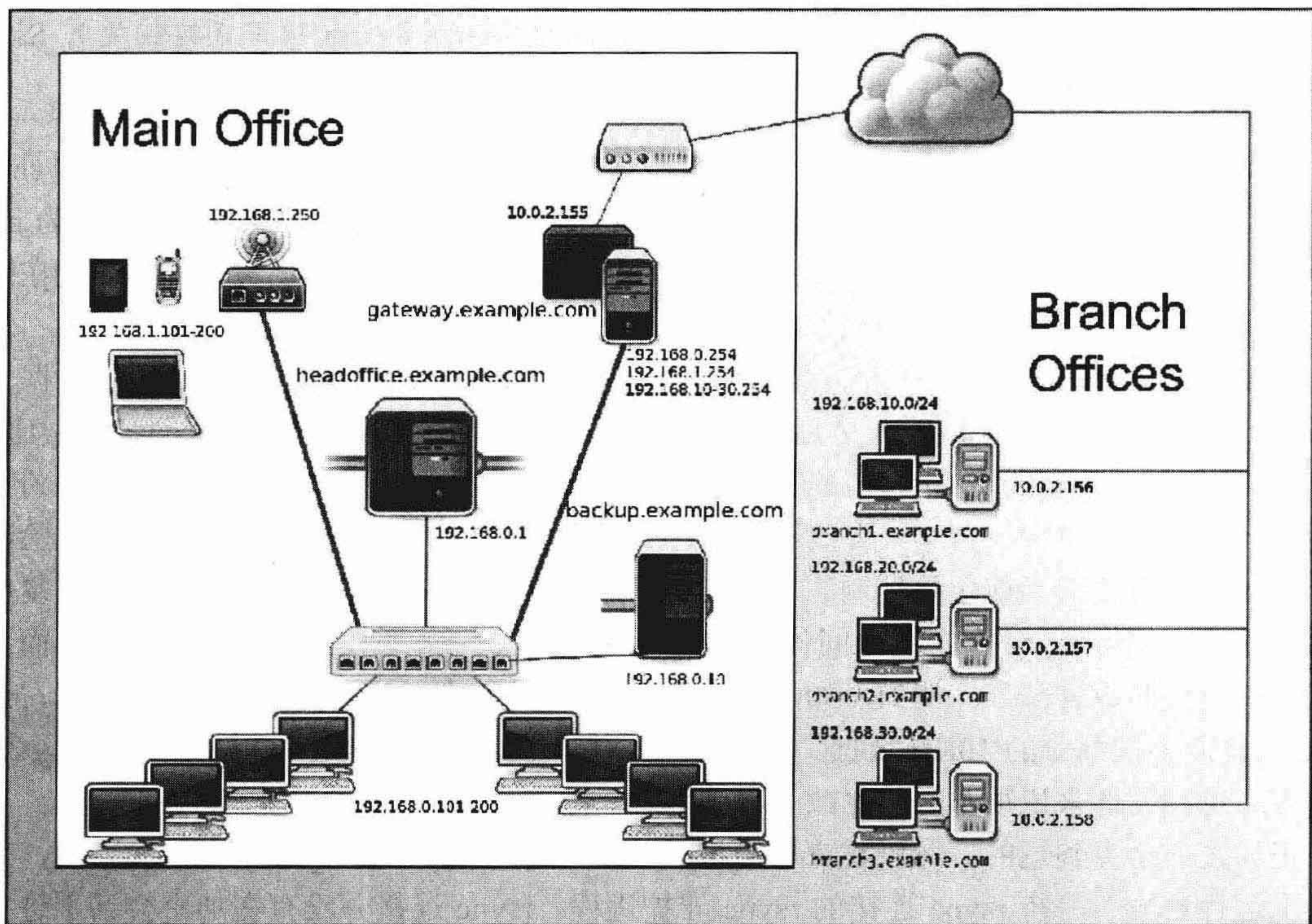


图 13-1 我们的网络

接下来向读者介绍备份上述网络的两种方法。把远程主机群上的数据复制到中心服务器 `headoffice.example.com`。在复制完这些远程文件之后，接着可以使用一个功能完备的备份应用程序备份这些数据，该备份应用程序位于主机 `backup.example.com` 上。

接着选用一台独立主机部署备份服务器。理想情况下，我们并不希望主服务器同时也是备份服务器，因为如果失去了主服务器，那么也会丧失恢复备份数据的能力。但是，如果没有能力多购买一台主机，考虑到有备份总好过没有备份，那就应该把数据备份到外部存储媒介上，例如 DVD 或数据磁带。

■注：传统意义上，Linux 系统的备份就是把一系列的命令手动编辑成脚本，这些命令包括 `tar`、`gzip`、`rsync`、`dd`、`cpio`、`dump`、`restore` 和 `mt`。应付一两台主机还可以这么做，但对于多台跨平台的主机来说这种方法就显得笨拙了。关于 Linux 系统众多的备份和恢复技术的阐述，感兴趣的读者可以查阅以下链接的内容，见“<http://www.ibm.com/developerworks/linux/library/l-roadmap8/>”。

13.4 使用 rsync

本节首先向读者介绍如何使用 `rsync` 工具安全高效地复制远程主机上的数据。这种方法适用于不太复杂、不要求备份计划面面俱到的网络。为此，准备使用 `rsync` 把一台主机的数据安全地复制到另一台主机。

■注：`rsync` 工具属于 Samba 产品包（与第 12 章提到的 Samba Server 项目直接相关）。Samba 社区已接手维护 `rsync`，它的主页是 <http://rsync.samba.org/>。

`rsync` 工具可以在“客户机/服务器”配置中或者远程 shell 中使用。在说到“远程 shell”的时候，意思是借助于另一种像 SSH 那样的传输机制在主机之间使用 `rsync`，让 SSH 为 `rsync` 程序提供目标主机的远程 shell（或者命令行）。正是因为这样使用 SSH，所以 `rsync` 能在因特网那样不安全的环境里安全高效地复制数据。

`rsync` 是一种快速高效的数据复制方式，因为它只复制与初始复制数据不同的数据。在 `rsync` 把远程数据初次复制到本机之后，后续的任何一次复制都要和本机已有的备份做比较。`rsync` 通过把所要复制的文件列表发送给接收主机实现上述比较。文件列表包括所要备份数据的路径名、权限、属主、大小、更改时间等，如果特别指定，还会包括数据文件的校验码。这样 `rsync` 就可以把当前所要复制的文件列表信息和初次备份的信息做比较，然后仅复制不同的文件，这使得 `rsync` 在主机之间传送数据非常高效。`rsync` 还校验待复制的文件，而仅复制文件中不同的数据部分。例如，如果一个 1MB 的文本文件含有 990KB 的相同数据，那么 `rsync` 仅复制不同部分的 10KB 数据。不幸的是，对二进制文件（如 JPEG 格式）无法这么做，因为对二进制文件来说很难区分更改了哪些部分。

正如之前所提到过的，可以在“客户机/服务器”模型中使用 `rsync`，在该模型里，目标主机上运行着一个等待 `rsync` 连接的 `rsync` 守护进程。`rsync` 连接并没有在加密会话上进行，因此可能存在安全问题，所以不打算进一步讨论这种 `rsync` 用法。本章准备讲述借助于 SSH 的 `rsync`，使用它来提供安全的数据传输。这也就意味着只需在防火墙上开启 TCP 22 端口，而不用开启 TCP 873 端口，该端口用于 `rsync` 守护进程。

读者可以从下列学习指南中读到更多关于设置 `rsync` 守护进程的信息。

- <http://sunsite.dk/info/guides/rsync/rsync-mirroring02.html>
- <http://everythinglinux.org/rsync/>

在 SSH 上使用 rsync

在因特网那样不安全的环境里对本地网络之外的远程主机进行备份的情况并不少见。在这种情况下，可以在 SSH 上使用 `rsync` 访问远程主机，并从远程主机上获取数据或者做备份。通过执行一个简单的 Bash 脚本，就可以自动化备份过程：连接远程主机，并把该远程主机上的文件复制到备份主机上。第一步就是使用无口令 SSH 密钥建立一个到远程主机的 SSH 连接。

第 3 章和第 9 章向读者介绍过 SSH 通信。通常，用一个合适的密码创建 SSH 密钥，就可以在登录远程主机的时候用该密钥来验证会话。用无口令 SSH 密钥建立连接是因为我们准备用 `crontab` 运行备份脚本。`crontab` 在第 5 章介绍过。所以，不需要任何的用户交互，就可以定期执行远程主机的备份操作。无口令密钥会造成安全上的隐患，所以稍后会采取几个步骤来避免密钥被滥用。

首先，创建一个新用户 `exbackup` 来管理备份。然后，生成 SSH 密钥。

```
$ sudo /usr/sbin/useradd -m -d /data/backups -u 525 -g adm exbackup
```

这里，创建了一个叫做 `exbackup` 的用户，该用户的 UID 为 525，其默认用户组为 `adm`。把该用户添加到 `adm`（或者称为管理员）用户组是因为 `adm` 用户组通常拥有读日志文件的权限。`exbackup` 用户的主目录是 `/data/backups`，这是存储所有备份的地方。接着，使用 `sudo` 命令登录该用户账号并创建 SSH 密钥。

```
$ sudo su - exbackup
$ mkdir .ssh && chmod 700 .ssh && cd .ssh
$ ssh-keygen -b 2048 -t rsa -f exbackup
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in exbackup.
Your public key has been saved in exbackup.pub.
The key fingerprint is:
18:33:26:40:78:65:b5:b1:a8:17:5b:b6:32:05:0a:a4 exbackup@backup-1.example.com
$ ls -l
total 16
-rw----- 1 exbackup adm 3239 Feb 21 10:07 exbackup
-rw-r--r-- 1 exbackup adm 752 Feb 21 10:07 exbackup.pub
```

首先，输入“`sudo su - exbackup`”命令，切换到 `exbackup` 用户的 shell 环境。然后创建 `.ssh` 目录用于存放密钥，如果该操作成功（如“&&”所示），则更改目录 `.ssh` 的权限并进入该目录，然后使用 `ssh-keygen` 命令创建密钥。该密钥的长度为 2048 字节，即 `-b 2048`，类型为 `rsa`（读者可以选择 `rsa` 或者 `dsa`，它们是用于密钥数字签名的加密算法），在此称其为 `exbackup`。在完成密钥创建时，会被要求提供密码，这个地方只是按两次回车键。可以看到在 `.ssh` 目录列出来的内容里有两个密钥，一个私钥和一个公钥，公钥由 `.pub` 后缀标识。把安全的私钥（`exbackup`）保存在这台主机上，而把公钥（`exbackup.pub`）的内容复制到远程主机上的 `exbackup` 用户的 `authorized_keys` 文件中。

现在需要稍微谈一下安全问题。通过因特网把无口令密钥传递到远程主机的做法具有潜在的安全风险，数据可以通过加密避免普通窥探，但是一个蓄意的攻击者可能利用密钥作为接入点攻击主机。当然也可以在远程主机上创建一种叫做 `chroot jail` 的安全机制，但是这会限制访问远程主机文件系统的某些部分。

■提示：想要获知关于如何建立 `chroot jail` 的信息，请访问“http://www.howtoforge.com/chroot_ssh_sftp_debian_etch_p2”或者“http://howtoforge.org/chroot_ssh_sftp_fedora7_p2”。

可以同时使用密码和密钥，用一个叫做 `key-chain` 的工具为当前的连接缓存密码，但是每

重启一次主机就必须重新输入密码，所以这么做并不理想。为了给攻击者增加难度，我们的做法是限制 SSH 密钥所支持的命令。我们在远程主机上创建一个叫做 `ssh_limiter.sh` 的可执行文件。当使用 SSH 密钥登录远程主机的时候，该脚本将被调用，并限制任何使用 SSH 密钥的用户只被允许执行一个命令。

通过往 `authorized_keys` 文件添加一些设置项，可以限制 SSH 所能进行的操作。`authorized_keys` 是远程主机上相应用户的 `.ssh` 目录下的一个文件，该文件含有用于连接远程主机的已授权的公钥副本。现在就在远程主机用户的主目录下创建它们。用于限制密钥操作的设置项包括列表 13-1 所列出的内容。

表 13-1

authorized_keys 设置项

设置项	描述
from	限制连接发起源。含有域、FQDN 和 IP 地址
command	指定一个命令，该命令在验证密钥的时候执行
environment	如果 sshd 支持，则设置用户环境
no-agent-forwarding	禁止 SSH 认证代理转发
no-port-forwarding	禁止 SSH 端口转发
no-X11-forwarding	禁止 X11 转发

表 13-1 列出了部分可用设置项。想要获得更多信息，参阅 `sshd` 的手册页面。

接下来用上述说明信息创建一个 `authorized_keys` 文件，并把它发送到远程主机。首先，把已有的 `exbackup.pub` 密钥复制到 `remote_authorized_keys` 文件中，以此创建一个含有 `exbackup` 用户公钥信息的文件。

```
$ cp exbackup.pub remote_authorized_keys
```

再创建一个叫做 `ssh_limiter.sh` 的文件，在使用 `exbackup` SSH 密钥连接远程主机的时候，强制连接命令执行该脚本文件，如列表 13-1 所示。

列表 13-1 限制使用密钥在 SSH 上所能执行的命令

```
$ vi ssh_limiter.sh
#!/bin/bash
# Command to be used by exbackup at example.com to limit what exbackup can
# do on a remote host.
# SSH2 stores the original command send by the remote host in a variable
# $SSH_ORIGINAL_COMMAND. We will use case to test and limit the commands
# we are running.

case "$SSH_ORIGINAL_COMMAND" in
  *\&*)
    echo "UNAUTHORIZED COMMAND"
    ;;
  *\;*)
    echo "UNAUTHORIZED COMMAND"
    ;;
  *\|*)
    echo "UNAUTHORIZED COMMAND"
    ;;
  *)
    ;;
esac
```



```
rsync\ --server*)
$SSH_ORIGINAL_COMMAND
;;
*)
echo "UNAUTHORIZED COMMAND"
;;
esac
```

在列表 13-1 中，用 Bash 脚本语言检验 SSH 用户当前所提交的命令。当对远程主机发起 SSH 连接的时候，变量 `$SSH_ORIGINAL_COMMAND` 存有该远程主机上执行的命令。那么如果发起以下 SSH 连接：

```
$ ssh somehost@example.com ls -l /tmp
```

变量 `$SSH_ORIGINAL_COMMAND` 的值就为 “`ls -l /tmp`”。当用 SSH 密钥提交该命令时，可以检验该变量，并判断其命令类型是否是 SSH 密钥可以使用的。在远程主机上执行 `rsync` 命令时，该变量含有 “`rsync --server<some other arguments for rsync>`” 命令，我们希望只允许执行该命令而拒绝其他命令。

`case` 语句检查变量 `$SSH_ORIGINAL_COMMAND`，确保它只含有命令 “`rsync -server`”。首先，拒绝控制型命令 `&`、`;` 和 `|`，这些命令可以用于在目标命令末尾添加其他命令。如果命令以 “`rsync -server`” 开头，则接受它（“`\ --server`” 确保 Bash 转义 “`<空格>--`” 语句）。其他传过来的命令可以通过 “`*`” 拒绝。`case` 语句以 `esac` 作为结尾。

现在，需要编辑 `remote_authorized_keys` 文件，为密钥添加一些设置项。

```
$ vi remote_authorized_keys
command="~/bin/ssh_limiter.sh",from="*.example.com",no-port-fowarding,no-X11-
forwarding,no-agent-forwarding ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAgEAp7jGL2il3QKREVTpNWkdPqiEbG4rKdCLt/nx57PHkZvz
SGI64Glsc10zIz92PBN/ZjNb4Z1ZaOGS7UYQOg4SHKXsw5/VHchIN1k3p9Vwm9rZUiDg3azKr9J+R
+r9TDhwReyYt0QhR/jlaZflgYS3+xRLs+bQb6UXVRrccygCFtxvrA2B5Kkgw2QJhctSlNRyi8XobUK
7kOs2Bw4zIY8hEZMRBFEibqi/diXPngWsMeo2UQQGICo6yXmgUKqiuQqlazdDuTbEstLS97/LdT
qWd9MNAsYk= exbackup@backup-1.example.com
```

这里，给 `remote_authorized_key` 文件添加了一些设置项，该文件最后将被放在远程主机上。指定在使用密钥的同时可以执行哪些命令，以及哪些主机可以使用该密钥发起连接，我们还限制了普通用户的常用功能。任何使用该密钥进行连接的用户现在仅可以执行 `ssh_limiter.sh` 脚本，该脚本只允许执行 `rsync` 命令，并且只允许连接发起方来自 `*.example.com` 域。如果需要，可以对 “`from=`” 设置项做更多限制，即设置连接发起方的主机 IP 地址。

通过相应的用户名和之前为用户 `exbackup` 所创建的目录镜像，开始配置远程主机。必须确保远程主机的 `exbackup` 用户的主目录下有一个 `.ssh` 目录，该目录的权限设置为 700。现在，以普通用户的身份把 `remote_authorized_keys` 文件复制到远程服务器。首先，把 `remote_authorized_keys` 文件复制到 `jsmith` 用户身份能访问的某个地方（`/tmp` 目录就可以），同时也把 `ssh_limiter.sh` 文件复制到 `/tmp` 目录下。

用户 `jsmith` 在远程主机上拥有管理员的 `sudo` 权限，以该用户身份作以下操作。

```
$ scp /tmp/remote_authorized_keys /tmp/ssh_limiter.sh
jsmith@branch1.example.com:~/
```

以上命令把 `remote_authorized_keys` 文件安全地复制到远程主机的用户 `jsmith` 的主目录。现在，输入下列 `ssh` 命令。


```
$ ssh -t jsmith@branch1.example.com 'sudo useradd -u 500 -g adm -m ➡  
-d /data/backups exbackup && sudo -u exbackup mkdir -p /data/backups/.ssh && ➡  
sudo chmod 700 /data/backups/.ssh'
```

上述命令在远程主机上创建 `exbackup` 用户，同时在 `exbackup` 用户的主目录下创建一个叫做 `.ssh` 的目录，并设置其权限为 700（这是出于 SSH 安全上的要求）。需要指出的是，因为用了“`sudo -u exbackup`”命令创建 `.ssh` 目录，所以该目录拥有正确的属主权限。双与号“`&&`”表示如果第一组命令执行成功，接着执行下一组的命令。在这个例子中，用“`ssh-t`”提交命令。`-t` 告诉 SSH 强制启用虚拟终端机（`pseudo-tty`），否则 `sudo` 命令将拒绝执行。一个虚拟终端机（`pseudo-tty`）就是当前运行的一个伪文本终端设备，因此 `sudo` 命令可以正确执行。

下一步，把 `remote_authorized_keys` 文件复制到目的地，顺便重命名该文件并设置它的权限。还要创建一个 `/data/backups/bin` 目录，并把 `ssh_limiter.sh` 文件移到该目录。

```
$ ssh -t jsmith@branch1.example.com ➡  
'sudo mv remote_authorized_keys /data/backups/.ssh/authorized_keys ➡  
&& sudo chown exbackup:adm /data/backups/.ssh/authorized_keys ➡  
&& sudo chmod 600 /data/backups/.ssh/authorized_keys ➡  
&& sudo -u exbackup mkdir /data/backups/bin ➡  
&& sudo mv ssh_limiter.sh /data/backups/bin ➡  
&& sudo chown exbackup:adm /data/backups/bin/ssh_limiter.sh ➡  
&& sudo chmod 750 /data/backups/bin/ssh_limiter.sh'
```

在上述代码中，使用反斜线（`\`）拆分命令行，反斜线告诉 Bash 解释器，命令在下一行继续，而不会在按下回车键之后立即执行当前代码行。就 Bash 而言，上述所有代码被视为一行。

接着在 SSH 下强制启用虚拟终端机，再次与分支主机 1 建立连接。`sudo` 命令创建的所有新文件和新目录的属主为 `root` 用户，除非在 `sudo` 命令之后加上“`-u <用户名>`”选项。因此，需要修改所创建的文件和目录的权限和属主。我们还要在 `/data/backups` 目录下创建一个 `bin` 目录，这是用户 `exbackup` 的主目录。在 `authorized_key` 文件里，指定了“`command=~/.bin/ssh_limiter.sh`”，所以 `ssh_limiter.sh` 脚本需要复制到 `bin` 目录下，并设置合适的权限。

下面准备在远程主机上创建一个文件，用以测试即将向读者介绍的 `rsync` 脚本。在远程主机上的 `/tmp` 目录下创建一个叫做 `/tmp/test_sync.txt` 的文本文件，并填上一些无用文本。

```
$ vi /tmp/test_sync.txt  
fldjfs1  
lfdsjfs1a  
fsdjfs1  
fjsdl  
fsjfs  
fsl  
fsa  
23433
```

上述命令在 `remotehost.example.com` 主机上创建了一个叫做 `/tmp/test_sync.txt` 的文件，并往这个文件里随意添加一些文本。如果读者用 `cat` 命令显示这个文件（`cat /tmp/test_sync.txt`）的内容，将会看到它包含的随意添加的所有文本。

现在，准备测试用 `rsync` 和 SSH 密钥备份位于远程主机上的这个文件。


```
$ sudo su - exbackup
[sudo] password for jsmith:
$ rsync -av -e 'ssh -i .ssh/exbackup' remotehost.example.com:/tmp/test_sync.txt /tmp
receiving file list ... done
test_sync.txt
sent 42 bytes received 194 bytes 472.00 bytes/sec
total size is 58 speedup is 0.25
$ cat /tmp/test_sync.txt
fldjfs1
lfdsjfs1a
fsdjfs1
fjsdl
fsjfs
fsl
fsa
23433
Ldjas
```

我们用 `rsync` 命令把文件 `test_sync.txt` 简单地复制到本地主机的 `/tmp` 目录下。通过使用 `cat` 命令显示该文件的副本内容，可以看到该文件已经被复制了。稍后讲解 `rsync` 命令的细节。

接下来往该文件多加几行文本并对它再做一次 `rsync` 操作。

```
$ vi /tmp/test_sync.txt
fldjfs1
...
<snip>
...
fsa
23433
ldjas
dfald
asd
12344556
```

然后保存该文件并再做一次 `rsync` 操作。

```
$ rsync -av -e 'ssh -i .ssh/exbackup' branch1.example.com:/tmp/test_sync.txt /tmp
receiving file list ... done
test_sync.txt
sent 48 bytes received 213 bytes 174.00 bytes/sec
total size is 77 speedup is 0.30
```

当对文件 `/tmp/test_sync.txt` 使用 `cat` 命令时，会看到该文件包含了原文件中新的变化。

```
$ cat /tmp/test_sync.txt
fldjfs1
...
<snip>
...
fsa
23433
ldjas
dfald
asd
12344556
```


一切正常，不需要用到密码就可以安全地同步远程主机上的某个文件。紧接着测试 `ssh_limiter.sh` 脚本，以确定它是否按预期设想生效。在这里，测试一下看是否可以使用密钥以 SSH 方式登录远程主机并执行 `top` 命令。

```
$ ssh -i .ssh/exbackup remotehost.example.com top
UNAUTHORIZED COMMAND
```

可以看到发送被禁止或不符合要求的命令将产生“UNAUTHORIZED COMMAND”回复。现在，可以创建 `crontab` 脚本，定期地把远程主机上的文件同步到本地主机的备份目录下。表 13-2 列出一些可以和 `rsync` 命令一起使用的参数。

表 13-2 rsync 命令参数

参数	描述
-a	通用归档模式，以递归方式复制文件，相当于-rlptgoD
-r	以递归方式复制目录及其子目录
-l	保留软连接
-p	保持文件权限
-t	保持文件时间信息
-g	保持文件属组权限
-o	保持文件属主权限
-D	保存设备（字符和块设备）和特殊文件（fifo 文件和 sockes 文件）
--exclude	忽略某些目录或文件，目录或文件名可写成字符串匹配模式。举个例子就是.svn/，表示忽略.svn/目录
--include	指定包括某些目录或者文件，调整待复制文件。与--exclude 的语法一样
-n, --dry-run	预览。显示将要发生的动作，实际上并不执行同步操作

通常需要结合归档参数集，也就是-a 来使用 `rsync` 命令。-a 是一个组合参数，代表参数组合“-rlptgoD”。这些参数是：-r，以递归方式拷贝；-l，保留软连接；-p，保持文件权限；-t，保持模块时间信息；-g，保持文件所属群组权限；-o，保持文件属主权限；还有-D，保存设备和特殊文件。这些参数通常足够用于对系统做归档操作了，不过如果需要，还可以添加更多的参数，这些参数在 `rsync` 手册页面上有解释。

我们还可能会用到--exclude 和--include 参数，这两个参数允许调整所要同步的文件或目录。还可以用“--exclude-from=<file>”和“--include-from=<file>”参数选定多个目标文件或目录。

接下来看一个典型的脚本，可以用它把远程主机的数据同步到本地主机。在远程主机上，有个需要同步到本地主机的目录/data/staff/sales，下面用备份程序对它做备份。有两个远程主机：branch1.example.com 和 branch2.example.com，我们准备使用 `rsync` 工具同步它们的 /data/staff/sales 目录中的内容，/data/staff/sales/temp 目录除外（这个目录要忽略掉的）。另外，需要以 `exbackup` 用户身份使用已经创建好的无口令密钥执行该脚本。

首先，创建一个.ssh/config 文件，该文件保存需要的所有 SSH 配置信息。在该文件中，添加主机名、IP 地址或全局域名（FDQN）以及用于连接的用户名。还要定义用于连接的 SSH 密钥。


```
$ cat .ssh/config
Host *.example.com
  User exbackup
  Identityfile ~/.ssh/exbackup
```

上述代码为每次 SSH 连接设置了用户名 `exbackup` 和 `~/.ssh/exbackup` 目录中的身份文件，还指定了所要连接的远程主机的域范围是 `example.com`。如此设置相当于执行命令 “`$ ssh -I ~/.ssh/exbackup exbackup@somehost.example.com`”。

用于在主机之间执行 rsync 操作的脚本如列表 13-2 所示。

列表 13-2 nightly_remote_sync.sh 脚本

```
1. #!/usr/bin/env bash
2.
3. # This uses rsync to sync down remote files to the /data/backups/<hostname>
4. # directories.
5. # The rsync command we will use.
6. RSYNC='which rsync'
7. RSYNC_OPTS="-av "
8.
9. # Host list - Bash array
10. HOSTLIST='
11. branch1.example.com
12. branch2.example.com
13. '
14. # Back up directory on local host and source directory on remote host
15. BACKUP_DIR='/data/backups/'
16. SALES_DIR='/data/staff/sales'
17.
18. # excluded directory
19. EXCLUDED="temp/"
20.
21. # error function
22. error_check() {
23.   if [ $1 -eq 0 ] ; then
24.     echo "backup successful"
25.   else
26.     echo "backup failed: see error number: $1"
27.   fi
28. }
29.
30. # The rsync functions
31. get_sales() {
32.   ${RSYNC} ${RSYNC_OPTS} --exclude $EXCLUDED $HOST:$SALES_DIR $BACKUP_DIR/$HOST ➡
33. }
34.
35. # Bash for loop to go through each host and rsync the data.
36. for HOST in $HOSTLIST ; do
37.   get_sales
38.   error_check $?
39. done
40.
41. exit 0
```


列表 13-2 中的脚本的目的是要把一个或多个远程主机上的文件同步到执行该脚本的主机的 /data/backups 目录下。执行脚本的主机可以是局域网内的备份主机 backup.example.com，我们将在该主机上安装合适的备份软件。不过，该方法只是众多备份方法中的一种。

下面讲解该脚本文件，第 1 行代码把执行环境设置为 bash 脚本，这里也可以使用传统语法（#!/bin/bash）让 Linux 系统知道我们准备执行 bash 脚本。

第 2~第 5 行是注释，用于描述该脚本。第 6、第 7 行设置 RSYNC 变量，第 10~第 13 行声明所要同步的主机列表，第 15~第 19 行是其他变量的声明。用来存放备份数据的备份目录是“BACKUP_DIR='/data/backups'”，业务目录“SALES_DIR='/data/staff/sales'”是备份的目标目录，temp/目录是要被忽略的目录。

■注：用户 exbackup 必须对所有要备份的文件和目录拥有读权限，并对存放备份的目录拥有写权限。可以考虑用组操作完成权限设置。

第 22~第 28 行是一个负责错误检查的 Bash 函数。如果脚本以非 0 代码结束，那么脚本执行失败；如果以代码 0 结束，则脚本执行成功。error_check()子程序或函数接收\$? 参数，也就是其他函数调用的结束代码，并测试该结束代码是否为 0。如果一切顺利，rsync 将以代码 0 结束，否则以其他代码结束。可以使用该错误检查函数检查其他函数执行成功还是失败，函数执行成功则返回 0。

第 31~第 33 行声明的 get_sales 函数描述了 rsync 的用法，即使用 rsync 命令把 /data/staff/sales 目录同步到/data/backups/<hostname>目录下。在第 32 行，“2>&1 >/dev/null”把标准输出和标准错误输出定向到/dev/null。注意/dev/null 是一个 Linux “黑洞”，如果发送数据给它——如标准输出数据和标准错误输出数据——这些数据将消失得无影无踪。

■注：在 Linux/UNIX 主机上运行一个程序时，有 3 个标准的特殊文件描述符被使用到，即 stdin（标准输入）、stdout（标准输出）和 stderr（标准错误输出）。当程序需要接收输入数据时，把输入端和 stdin 文件描述符绑定就能接收数据了。同样道理，当程序产生输出数据时，就会把输出数据写到 stdout。如果程序产生某个错误，就把该错误信息写到 stderr。要获得更多关于使用和重定向 stdin、stdout 及 stderr 的信息，请访问“<http://learnlinux.tsf.org.za/courses/build/shell-scripting/ch01s04.html>”。

最后，第 36~第 39 行代码使用循环语句遍历主机列表中的每台主机，并对每台主机进行 rsync 操作。然后检查每次 rsync 操作后的错误代码，查看该代码是否为 0，如果为 0 则打印出成功信息。

■注：想要了解更多关于 bash 编程的知识，“<http://www.tldp.org/LDP/abs/html/index.html>”便是个很好的资源。另外，SanderVan Vugt 写的书《Beginning the Linux Command Line》（Apress, 2009）也是很好的相关资料。

在命令行下执行该脚本时，得到以下结果。

```
exbackup@au-mel-ubuntu-1:~$ ./bin/nightly_remote_rsync.sh
backup successful
backup successful
```


■注：在首次执行该脚本的时候，SSH 会要求确认远程主机的新密钥签名。键入“yes”确认密钥签名，从此以后脚本就会正常执行。

现在，把该脚本写入 `crontab` 文件，这样就可以定期地执行该脚本了。回忆第 5 章讨论过的 `crontab` 文件，然后创建一个 `/etc/cron.d/example_nightly_sync` 文件，并添加以下内容。

```
# run the nightly rsync script at 5 minutes past 12 every morning.
MAILTO=jsmith@example.com
5 0 * * * exbackup ➡
/data/backups/bin/nightly_remote_rsync.sh
```

这里，每天晚上 0 点 5 分以 `exbackup` 用户身份执行一次该脚本文件。如果出错，则发送一封邮件到 `jsmith@example.com`。对于把一台主机上的文件同步到另一台主机来说，这虽然是一个好方法，但实际上不算是一种很好的备份策略。例如，每天半夜同步所有被修改过的文件，包括错误文件，如果这些错误文件不被及早发现，那么它们就会被传送到备份主机，于是就会失去有用的数据副本。当采用一种合适的备份策略时，那些会使我们丧失恢复数据能力的潜在威胁应该被减少到最小。

一些围绕 `rsync` 的备份工具已经被开发出来，如下所示。

- BackupPC: <http://backuppc.sourceforge.net/>
- drsync: <http://hacks.dlux.hu/drsync/>
- Dirvish: <http://www.dirvish.org/>
- SystemImager: http://wiki.systemimager.org/index.php/Main_Page

接下来，介绍一个名叫 `Bacula` 的开源备份服务器应用程序，该备份服务器应用程序能够备份 Linux 系统和 Windows 系统上的数据。

13.5 使用 Bacula

商业的备份解决方案的价格可能是极其昂贵的，但是 `Bacula` 却能为 Linux、Unix 及 Windows 的桌面系统和服务器提供健壮的、可信赖的、可定制的和高效的开源备份服务。`Bacula` 支持大多数存储设备，如 DAT、LTO、和自动加载器（autoloader），当然它还可以把数据备份到磁盘。`Bacula` 易于配置、维护和升级，是个完整的、免费的、健壮的备份服务器。

`Bacula` 按照客户机/服务器模型工作，备份的目标主机需要安装 `Bacula` 客户端。`Bacula` 服务器本身需要运行两个守护进程：Director 守护进程和 Storage 守护进程。

Director 守护进程管理备份对象、备份时间和备份存储地点，它还为任何需要数据恢复的任务提供类似的服务。Director 守护进程需要一个配置文件配置其自身运行的详细信息，Storage 守护进程和 File 守护进程也有类似的配置文件，不过 File 守护进程运行在目标主机上。

Storage 守护进程分别和 Director 守护进程还有运行在目标主机上的 File 守护进程进行通信，并控制对数据存储设备（包括磁盘和磁带）的访问。Storage 守护进程控制备份存储

媒介、磁带驱动器和自动加载器的访问。经过配置，Storage 守护进程甚至可以把备份数据写入 DVD。

File 守护进程运行在目标主机上，等待 Bacula 的 Director 守护进程发起连接。当 File 守护进程收到启动备份的指令时，目标主机就会获得所要备份的文件列表。然后，File 守护进程直接连接 Storage 守护进程，并把备份数据发送过去，这些备份数据将被写到备份的存储媒介上。File 守护进程会把已经备份好的文件信息发送给 Director 守护进程，然后这些信息被写到 Bacula 的任务日志里。

Bacula 的任务日志记录已备份文件信息、备份存储位置、备份时间和备份存储媒介（或卷）。该任务日志保存在 SQL 数据库中，以备日后查询。一旦备份完成，Bacula 程序通过比较写入任务日志的备份信息和写入磁带的备份信息，验证备份是否成功。

当需要进行恢复操作时，Bacula 读取任务日志中的内容，请求需要载入合适的存储媒介。然后 Bacula 通知目标主机和 Storage 守护进程，于是它们启动恢复过程。

在 Bacula 服务器里，备份和恢复操作被称为任务（job）。可以在 Bacula 的 Director 配置文件（/etc/bacula/bacula-dir.conf）里安排备份和恢复任务。每个任务由一系列的设置组成，这些设置可以继承于一组通用的设置。每个任务在客户机（也就是目标主机）上执行，它们把数据备份到一个卷（volume）或者从一个卷（volume）中恢复数据。这里的“卷”指的是所使用的存储媒介（例如磁带、DVD 或者磁盘）的部分。多个卷可以组合成池（pool），还可以给这些卷附加一些通用设置。这些设置与保存周期、用途和是否循环覆盖相关。当然，备份安排人员可以通过协同管理任务、客户机、卷以及池进行一次性操作或者重复性操作。

可以通过 bconsole 控制 Bacula 的操作，bconsole 是一个终端控制台程序，它可以用来执行备份任务以及显示 Director 守护进程、Storage 守护进程以及 File 守护进程的状态信息。它还可以用来管理卷、池和恢复操作。bconsole 安装简单，可以被安装到网络的任何地方。

■注：在“介绍 Bat 控制台”章节，将介绍 Bacula 的 GUI 控制台—Bat 的安装。

Bacula 服务器要求开启下列 TCP 端口。

- Bacula Director 守护进程要求开启 9101 端口（仅在 Bacula 服务器上开启）。
- Bacula Storage 守护进程要求开启 9102 端口（仅在 Bacula 服务器上开启）。
- Bacula File 守护进程要求开启 9103 端口（在所有目标主机或客户机上开启）。

在备份服务器上使用下列命令添加上述端口。

```
$ sudo /sbin/iptables -I Firewall-eth1-INPUT <rule-number> -p tcp -m state --state NEW -m tcp --dport 9101:9103 -j ACCEPT
```

在客户机上，需要开启 TCP 9103 端口。

13.5.1 获取软件

SourceForge 网站的 Bacula 页面（<http://sourceforge.net/projects/bacula>）上提供最新的

release 版本的 rpm 包和 deb 包供读者下载。大部分的发布版本比最新的 release 版本还要新。可以在以下页面下载 Bacula 的软件包和压缩包：

http://sourceforge.net/project/showfiles.php?group_id=50727。

可以用常用的软件包安装工具在 Red Hat 和 Ubuntu 主机上安装 Bacula。Fedora FC8 以上版本则要进行以下操作。

```
$ sudo yum install bacula-client bacula-director bacula-storage bacula-console
```

对于 RHEL (Red Hat Enterprise Linux, 红帽企业版) 主机, 需要从 SourceForge 上下载安装包。http://sourceforge.net/project/showfiles.php?group_id=50727&package_id=213714 有一些免费的 RPM 包。还要确保主机上装有 MySQL 数据库服务器, 但是, 如果想用其他主机上的 MySQL 数据库服务器, 则不必使用本机上的。Bacula 和 MySQL 数据库的安装包均可以通过 RHN (Red Hat Network, 红帽网络) 获得, 我们需要下载下列软件包。

```
bacula-mysql-2.4.2-1.el5.i386.rpm bacula-mtx-2.4.2-1.el5.i386.rpm
```

确保已经创建了目录 `/var/spool/bacula`, 如果该目录不存在, 使用 `mkdir` 命令创建它。

在 Ubuntu 系统上, 必须执行下列命令。

```
$ sudo aptitude install bacula-fd bacula-console bacula-director-mysql
bacula-sd-mysql
```

按要求提供 MySQL 服务的用户名和密码。上述命令会安装所有必需的软件包和依赖软件, 并在配置目录 `/etc/bacula` 下创建默认的配置文件。用户 `bacula` 也在此时被创建。在启动和运行 Bacula 之前, 必须编辑位于配置目录下的下列文件来访问 Bacula 管理控制台: `/etc/bacula/bacula-dir.conf`、`/etc/bacula/bacula-sd.conf`、`/etc/bacula/bacula-fd.conf` 和 `/etc/bacula/bconsole.conf`。

对 Bacula 做任何配置之前, 需要解释一下这些配置文件之间的相互关系。由于 Bacula 守护进程相互通信, 因此它们必须交换密码。每一个 Storage 守护进程 (可以有多个) 和每一个 File 守护进程 (我们通常有多个) 必须拥有 Bacula Director 守护进程的名字和用于自身验证的密码。对于每一个要和 Director 守护进程通信的守护进程, 其密码都被设置在两个地方。

图 13-2 显示了配置文件及其通用设置之间的关系。

如图 13-2 所示, Director 守护进程的 Bacula 配置文件 `/etc/bacula/bacula-dir.conf` 中包含 Storage 守护进程、客户端守护进程以及 `bconsole` 程序的信息。还可以看到各种配置对象的定义, 例如 Storage、Pools、FileSets。这些定义必须设置对象名字, 还必须包括在尖括号中, 即 `{<definition = value>}`, 而且每行只能有一个设置。

可以从图 13-2 中看到, Storage 文件中的 Name 与 Media Type 必须匹配 Director 文件中 Device 对象定义中的 Device Type 与 Media Type。Storage 文件中 Director 对象定义中的 Name 与 Password 必须匹配 Director 文件中 Storage 片段的 Name 与 Password。

在 Director 文件上定义一个客户端时, 其 Password 赋值必须等于 File 守护进程的配置文件中的与 Director 名字对应的 Password 的值。在图 13-2 中还会看到, `bconsole` 程序为了能够与 Bacula 建立连接, 其配置文件必须含有 Director 的名字和密码。

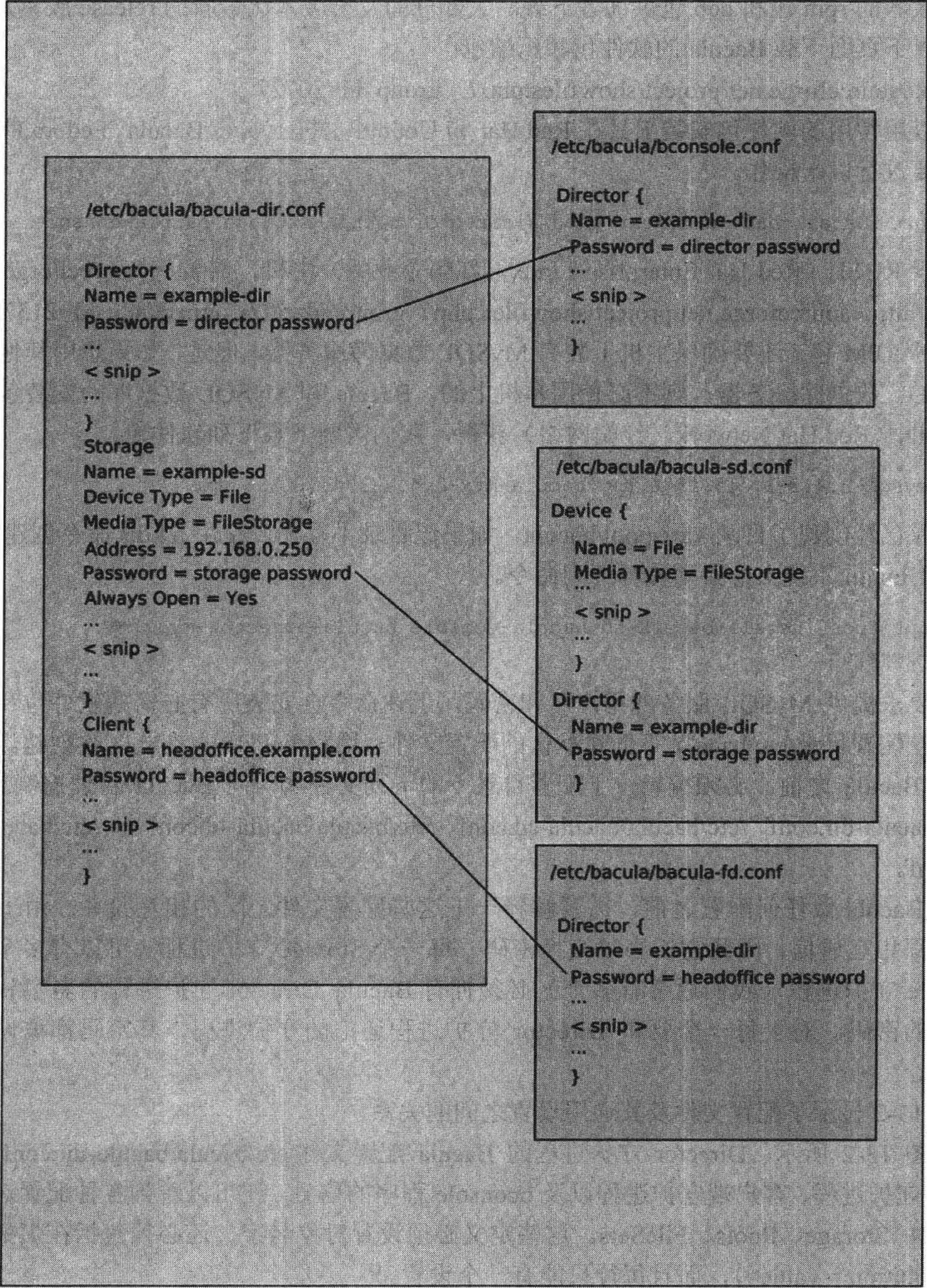


图 13-2 Bacula 通用配置文件的设置

第 11 章讨论过 MySQL 服务器。现在准备建立一个数据库用以存放任务日志，同时配置用以备份的 MySQL 主机。首先，需要在 MySQL 主机上创建正确的账号和数据库。

假设主机 headoffice.example.com 上已经安装好了 MySQL 数据库服务器，然后我们准备执行主机 backup.example.com 上的备份任务。如果没有另外一台可用主机充当独立的备份主

机，就可以把 MySQL 服务器安装到同一台主机 `headoffice.example.com` 上。在这种情况下，下面的例子中用到 `backup.example.com` 主机的地方就要替换成本地主机。另外一种替代方案是在备份主机上安装一个独立的 MySQL 数据库服务器、PostgreSQL 数据库服务器或者 SQLite 数据库服务器。

在 MySQL 主机 `headoffice.example.com` 上，创建一个新的 Bacula 数据库，并授予用户 `bacula` 访问权限。然后通过改变防火墙，确保 `backup.example.com` 主机能够访问 `headoffice.example.com` 主机上的 3306 端口。与此同时，还要确保 Bacula File 守护进程已经安装在 `headoffice.example.com` 主机上，并且可以穿过防火墙访问 `backup.example.com` 主机上的 TCP 9103 端口。

在主机 `headoffice.example.com` 上，进行以下操作。

```
$ sudo mysql -u root -p
Enter Password:
...
mysql> CREATE DATABASE 'bacula';
Query OK, 1 row affected (0.00 sec)
mysql> GRANT ALL PRIVILEGES ON bacula.* to bacula@backup.example.com ➡
IDENTIFIED BY 'somepassword';
Query OK, 0 rows affected (0.00 sec)
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
mysql> \q;
Bye
```

对于 Red Hat 主机，可以在 `/usr/lib/bacula/mysql` 目录中获得创建数据库的 SQL 语句；而对于 Ubuntu 主机，相应的 SQL 语句则位于 `/usr/share/bacula-director` 目录中。

为了给 Bacula 数据库建表，需要在 `backup.example.com` 主机上执行以下操作。

```
$ sudo -u bacula /usr/lib/bacula/make_mysql_tables -h headoffice.example.com ➡
-u bacula -p
Creation of Bacula MySQL tables succeeded.
```

执行 Bacula 所提供的 SQL 脚本将创建必需的数据表。在同一目录下，还会发现可以使用类似的脚本创建数据库，并分配所需要的有效权限。

执行下列命令创建该数据库。

```
$ sudo -u bacula mysql -h headoffice.example.com -u bacula -p -D bacula ➡
-e 'show tables;'
```

该命令用 `bacula` 用户的用户名和密码连接 `headoffice.example.com` 主机的 MySQL 数据库。“-p”提示输入密码，“-psomepassword”就是命令行下提交密码的格式（表示“-p”与密码之间没有空格）。命令行的输出信息将说明两点，确认用户名与密码正确及 SQL 脚本成功创建一些数据库表。

■注：访问网址“http://www.bacula.org/en/rel-manual/Installing_Configurin_MySQL.html”，可以找到更多关于 Bacula 与 MySQL 联合使用的信息。

接下来必须确保 `/var/spool/bacula` 目录存在，并且该目录对用户 `bacula` 可写。其他需要注意的是，`/data/backups/bacula/FileStorage` 目录必须存在，并且对用户 `bacula` 也必须是可写的，`/var/log/bacula.log` 文件也必须存在并对用户 `bacula` 可写。

13.5.2 配置 Bacula

现在查看 Bacula 服务器必备的配置文件。如先前所述，Bacula 服务器本身有 3 个配置文件，还有一个管理控制台的配置文件。首先，看一下主配置文件，也就是 Director 守护进程的配置文件/etc/bacula/bacula-dir.conf。

1. bacula-dir.conf

Director 配置文件是 Bacula 使用的主配置文件。该配置文件定义 Bacula 服务的各个方面：任务（jobs）、调度（schedules）和池（pools）等。

“Director {}”中的内容定义了 Director 守护进程本身。

```
Director {                                # define myself
    Name = example-dir
    DIRport = 9101                        # where we listen for UA connections
    QueryFile = "/etc/bacula/query.sql"
    WorkingDirectory = "/var/spool/bacula"
    PidDirectory = "/var/run"
    Maximum Concurrent Jobs = 1
    Password = "directorpassword" # Console password
    Messages = Daemon
}
```

在上述定义中，只需要修改 Name 和 Password 设置，其余字段保持默认值就可以了。Director 守护进程的端口被设置为 9101。QueryFile 直接指向/etc/bacula/query.sql 文件。/etc/bacula/query.sql 是一个 SQL 文件，它包含各种用于查询数据库的 SQL 语句。如果认为 Bacula 无法提供要查询的信息，则可以往该脚本文件添加自己的 SQL 查询语句。

现在看 JobDefs 的定义，默认的 Job 或者 JobDefs 定义可以把相似的备份任务需求组合在一起。例如，多个 Web 服务器可以组合成一个 JobDefs，该 JobDefs 定义这些 Web 服务器的 FileSet、Schedule 和 Storage 等属性。而多个邮件服务器则可以定义成另一种 JobDefs。Job 的所有属性都可以在 JobDefs 中定义。

```
JobDefs {
    Name = "DefaultLinux"
    Type = Backup
    Level = Incremental
    Client = headoffice-fd
    FileSet = "Linux"
    Schedule = "WeeklyCycle"
    Storage = File
    Messages = Standard
    Pool = Full
    Full Backup Pool = Full
    Incremental Backup Pool = Incremental
    Priority = 10
}
```

上述代码取自文件/etc/bacula/bacula-dir.conf，里面用到的所有属性均可在 Job 资源中定

义。首先，在一对大括号“{}”之间声明一个 JobDefs。接着，把该 JobDefs 命名为 DefaultLinux。任务类型 Type 可以是 Backup、Restore、Verify 或者 Admin 中的一种。Backup 任务类型表示执行备份任务，Restore 类型表示执行恢复任务，Verify 类型表示把任务日志里的内容和已备份好的文件集合作比较，而 Admin 类型表示在需要时可以执行任务日志的维护操作。

备份任务需要的 Level 属性实际上不是必需的，但该属性必须定义在 JobDefs 或者 Schedule 中。在 Schedule 中对该属性所做的任何设置都会覆盖掉在 JobDefs 中对该属性的设置。Level 值可以是 Full、Incremental 或 Differential。完全（Full）备份意味着对 FileSet 所指定的文件集合的所有文件做备份，而不考虑这些文件的备份历史。增量（Incremental）备份意味着对在上一次完全备份之后被修改过的文件部分做备份。差异（Differential）备份所做的工作与增量备份非常相似，但它对在上一次完全备份之后的被修改的所有文件做备份。

每当执行一次差异备份或者增量备份，Bacula 的 File 守护进程客户端就会查看备份文件列表，得到 modify 属性发生变化的文件（modified files）的时间戳，还有 change 属性发生变化的文件（changed files）的时间戳，并把二者与上一次完全备份的文件信息做比较。如果前后的时间戳不一致，那么就通过差异备份或增量备份的过程对这些时间戳不一致的文件做备份。换言之，差异备份与增量备份的不同之处就是，差异备份把最后一次完全备份以来所有发生变化的文件做成一个备份。这意味着虽然差异备份会在我们的卷（volume，存储媒介）上占用更多的空间，但是要从备份中恢复数据时，只需要最后一次完全备份和最近一次差异备份就足够了。如果采用的是增量备份，则需要最后一次完全备份加上该次完全备份之后的每一次增量备份。这里需要说明的另外一点是，对于前面提到的客户机上的 FileSet 表示的数据，如果没有为它指定具有一定保存周期的完全备份，Bacula 将自动地把它的备份类型升级到完全备份。在本例子的 JobDefs 定义里，我们设置“Level = Incremental”。

在 JobDefs 的定义里，Client 属性被设置为 bacula-fd。该属性会在“Client {}”的定义中被覆盖，但它在当前的配置文件中是必需的。客户端的 bacula-fd 资源必须定义在配置文件中，否则会发生错误。我们在本例子里保持其默认定义不变。

“FileSet = "Linux"”指令定义了描述所要备份的文件的资源。FileSet 属性将在 Director 配置文件中做进一步设置。在 FileSet 的设置里，可以包括和排除一些目录、文件名或文件类型，还可以指定包括 Linux 和 Windows 文件在内的哪些文件需要备份，哪些将被排除。在本例子里，默认的文件集所包含的内容是一个典型的 Linux 文件集合。

接下来要给前面的 JobDefs 指定 Schedule 资源，Schedule 规定何时执行某种类型的任务。在 Bacula 中，WeeklyCycle 属性的默认设置是每周做一次差异备份，每月做一次简单的完全备份，并且从周一到周六每天半夜做一次增量备份。我们准备修改这种备份策略，使得每隔三个星期才做一次完全备份，稍后就会谈到如何修改。

在默认的文件集定义里，可以看到 Storage 资源被设置成 File。这里的 File 指的是存储设备配置文件里的名字，即“Name = File”。之前解释过，该 Storage 指向 Storage 守护进程的配置文件，备份数据将被存储到预设的硬盘上，即/data/backups/bacula/FileStorage。

“Messages = Standard”设置项与 Messages 资源相关。通过该属性，可以设置消息的处理方式。这些消息可以被写到日志里，也可以被发送到某个电子邮箱地址。

“Pool = Full”设置项为备份任务指定一种特定的、由多个卷（volume）组成的池，该池

用于存储备份。正如前面所提到的，卷就是存储媒介的组合。需要一组存储媒介用于每天一次的备份，一组用于每周一次的备份，还有一组用于每月一次的备份。本例为备份设置了默认的存储媒介组：Full。并且，设置“Full Backup Pool = Full”和“Incremental Backup Pool = Incremental”，以确保完全备份和增量备份的数据被存储到正确的媒介上。

在 JobDefs 定义的最后，设置“Priority = 10”。该设置用以分配任务优先级。Priority 数值越小，Bacula 分配给该任务的优先级就越高。JobDefs 定义中的某些设置可以在其他一些定义中重置，例如在 Job、Schedule 以及 Client 定义中重置 JobDefs 的某些设置。

```
Job {
    Name = "BackupCatalog"
    JobDefs = "DefaultLinux"
    Level = Full
    FileSet="Catalog"
    Schedule = "WeeklyCycleAfterBackup"
    # This creates an ASCII copy of the catalog
    RunBeforeJob = "/usr/libexec/bacula/make_catalog_backup bacula bacula"
    # This deletes the copy of the catalog
    RunAfterJob = "/usr/libexec/bacula/delete_catalog_backup"
    Write Bootstrap = "/var/spool/bacula/BackupCatalog.bsr"
    Priority = 11 # run after main backup
}
Job {
    Name = "RestoreFiles"
    Type = Restore
    Client = headoffice-fd
    FileSet = "Linux"
    Storage = File
    Pool = Default
    Messages = Standard
    Where = /tmp/bacula-restores
}
Job {
    Name = headoffice.example.com
    Client = headoffice-fd
    Enabled = yes
    JobDefs = "DefaultLinux"
}
```

前两个 Job 定义是标准的 Bacula 任务定义。第一个任务叫做 BackupCatalog，它用于备份存有任务日志的 MySQL 数据库。第二个任务定义用于恢复任务。在 headoffice.example.com 任务定义里，声明 Name 和 Client（稍后定义它们），设置“Enabled = yes”激活任务，并给它设置一个 JobDefs。大部分能被分配到任务资源定义中的设置项都是通过“JobDefs = \"DefaultLinux\"”被定义的。

剩下需要明确定义的包括 FileSet、Pool 和 Schedule，此外还要设置一个 Client。首先观察 FileSet 定义，FileSet 指明要在主机上备份的内容。如果需要，可以声明多个 FileSet，而且既可以声明 Linux 系统的文件集合也可以声明 Windows 系统的文件集合。接下来要创建一个叫做 Linux 的 FileSet。


```

FileSet {
  Name = "Linux"
  Include {
    Options {
      Compression = GZIP
      Signature = SHA1
    }
    File = "/etc"
    File = "/var/lib"
    File = "/data"
    File = "/home"
  }
  Exclude {
    File = "/proc"
    File = "/sys"
    File = "/dev"
    File = "/data/backups/bacula/FileStorage"
  }
}

```

在这个 FileSet 定义里，首先设置 Name 属性。FileSet 定义可用于 JobDefs 或者 Job 的定义。正如我们所看到的，可以把 FileSet 分成两个部分——Include 部分和 Exclude 部分。在 Include 部分里，指定了一对 Options——Compression 和 Signature。这两项设置允许使用 gzip 程序压缩备份数据，并设置数字签名算法为 SHA1 或 MD5。当备份数据被写到磁带时，如果使用磁带机对备份数据做硬件压缩，那么就不再对数据做软件压缩了。关于数字签名算法，可以选择 SHA1 或 MD5，但是 MD5 会在数据大小和 CPU 时间方面增加额外的开销。数字签名信息被保存在任务日志里，用于验证文件内容是否被修改。接着指定想要备份的文件，它们是 /etc、/var/lib、/data 和 /home 等目录。选择 /etc 是因为该目录包含系统的主要配置文件，选择 /var/lib 是因为 Samba 和 MySQL 的一些数据存储在该目录下，选择 /data 和 /home 是因为它们是存储用户数据的地方。

另外需要排除一些目录（或者至少保证它们不会被错误地备份），即 /proc、/sys 和 /dev。这些目录是临时性的，它们在系统每次启动时会发生改变，所以保存它们没有意义。/data/backups/bacula/FileStorage 目录被排除在外是因为它是存放备份的地方。在这里对备份数据再做备份是没有意义的，如果这么做了，每一个备份的大小将会翻倍。

备份文件列表可以保存在一个独立的文件中，在 FileSet 设置中用 @ 符号标明该文件。例如，假设有一个 Windows 文件列表如下所示，已经把该文件列表保存到一个叫做 /etc/bacula/windows.list 的文件里。

```

File = "C:/"
File = "D:/"

```

在 FileSet 设置中，按以下方法把该文件包括进去。

```

FileSet {
  Name = "Windows complete"
  Include {
    Options {
      signature = SHA1
    }
  }
}

```



```
}
@/etc/bacula/windows.list
Options {
    Exclude = yes
    WildDir = "C:/WINNT/TEMP"
    WildDir = "C:/winnt/temp"
    WildDir = "C:/WINDOWS/TEMP"
    WildDir = "C:/windows/temp"
    WildDir = "**/Cache/*"
    WildDir = "**/cache/*"
}
}
```

这里，通过指定“@/etc/bacula/windows.list”文件，把 Windows 主机上的 C 盘和 D 盘包括进来。同时指定任务日志使用的数字签名算法为 SHA1。当然，还要排除一些不必要的目录。不需要备份临时目录或缓存目录，因为这些目录常常存有大量的数据，这些数据并不需要备份而且还十分庞大。

■注：关于如何在 Windows 平台下用完整的 Visual SourceSafe 支持使用 Bacula，读者可以在“http://www.bacula.org/en/rel-manual/Windows_Version_Bacula.html”上获得相关信息。

```
FileSet {
    Name = "Catalog"
    Include {
        Options {
            signature = MD5
        }
        File = /var/spool/bacula/bacula.sql
    }
}
```

最后的一个 FileSet 在 BackupCatalog 任务中被引用。接下来，讨论 Schedule 资源。Schedule 在需要运行任务的时候定义，还可以指定备份任务的级别（如果在 Schedule 中设置了任务的级别，该设置就会覆盖 Job 或者 JobDefs 中任务级别的设置）。对于将要用到的叫做 WeeklyCycle 的 Schedule，要对它的默认设置做点改动。我们希望每半个月做一次完全备份，因此把该 Schedule 配置成如下所示。

```
Schedule {
    Name = "WeeklyCycle"
    Run = Full 1st 3rd sun at 23:05
    Run = Differential 2nd 4th 5th sun at 23:05
    Run = Incremental mon-sat at 23:05
}
Schedule {
    Name = "WeeklyCycleAfterBackup"
    Run = Full sun-sat at 23:10
}
```

正如上述名叫 WeeklyCycle 的 Schedule 所表示的那样，在每个月第 1 个周六和第 3 个

周六做一次完全备份，每月的第 2、4、5 个周日做一次差异备份，并且在每周从周一到周六的半夜做一次增量备份。这些“日期-时间”规范由数个意义明确的关键字组成，如果需要，还可以制定出复杂的备份时间安排。

■注：关于能够在 Schedule 资源文件中使用的关键字，读者可以在“http://www.bacula.org/en/rel-manual/Configuring_Director.html”上查阅到更多信息。

在执行任何备份任务之前，必须创建 Client 资源。通过指定 Address、Name 和 Password，创建 Client。该 Client 定义中的 Name 必须匹配 Job 定义中设置的 Client，Password 必须匹配 File 守护进程配置文件中设置的密码。Address 可以是全局域名（fully qualified domain name）或者是 IP 地址。该 Client 资源如下所示。

```
Client {
  Name = headoffice-fd
  Address = headoffice.example.com
  Password = "headofficepassword"
  Catalog = MyCatalog
}
```

现在，配置 Storage 定义。再次强调，该定义将包含在 bacula-dir.conf 文件的“Storage {}”定义里。稍后在 bacula-sd.conf 文件中，会用到这一信息。

```
Storage {
  Name = File
  Address = backup.example.com # 注意：这里须用到全限定域名
  SDPort = 9103
  Password = "storagepassword"
  Device = FileStorage
  Media Type = File
}
```

以上定义是 Storage 的默认配置，只改变了 Password 和 Address 的信息。在图 13-1 中，会看到 Name、Password、Device 和 Media Type 等信息也出现在/etc/bacula/bacula-sd.conf 文件中。

接下来，把这些信息添加到 Director 配置文件中。任务日志的配置从 Catalog 定义开始。

```
Catalog {
  Name = MyCatalog
  dbname = bacula;
  user = bacula;
  password = "somepassword";
  DB Address = headoffice.example.com
}
```

Name 可以是任意字符串，我们使用默认的 MyCatalog。dbname 必须是 bacula，user 也必须是 bacula。password 就是之前设定的密码。DB Address 必须是该 Catalog 所在主机的地址。

如之前所述，卷池（Pool）就是组合在一起发挥效用的多个存储媒介卷。由一组存储媒介形成的池可用于按月、按周、按日进行的备份，或者用于归档操作，甚至用于 Web 服务器、

数据库服务器等。在 Pool 定义里，设置的内容包括保存周期策略、标记格式等。我们准备使用硬盘作为存储设备，所以存储媒介就不会有任何物理上的差别，不像使用磁带之类的媒介，需要载入或载出数据。在这里主要关心的是保存周期，换言之就是在删除备份数据之前要保存它多长时间。

Pool 定义如下所示。

```
Pool {
    Name = Default
    Pool Type = Backup
    Recycle = yes
    AutoPrune = yes
    Volume Retention = 365 days
}
Pool {
    Name = Full
    Pool Type = Backup
    Storage = File
    Maximum Volume Bytes = 500M
    AutoPrune = Yes
    Volume Retention = 6 months
    Recycle = Yes
    Recycle Oldest Volume = Yes
    Label Format = "Full-${Year}-${Month:p/2/0/r}-${Day:p/2/0/r}"
}
Pool {
    Name = Incremental
    Pool Type = Backup
    Storage = File
    Maximum Volume Bytes = 150M
    AutoPrune = Yes
    Volume Retention = 14 days
    Recycle = Yes
    Recycle Oldest Volume = Yes
    Label Format = "Incremental-${Year}-${Month:p/2/0/r}-${Day:p/2/0/r} "
```

以上代码定义了两个池，它们都使用同一个叫做 File 的 Storage。Pool 的 Name 属性可以被其他资源定义中的 Pool 属性引用。每个 Pool 定义都设置其 Pool Type 属性为 Backup，因为根据 Bacula 文档说明，Backup 是目前唯一可用的类型。设置两个不同的 Maximum Volume Bytes，一个为 500MB，一个为 150MB。当备份所需卷的大小达到上述限制时，就会转而请求另一个大小合适的卷（或者创建一个新卷）。这意味着应该备份多个小的数据文件，而不是备份一个囊括所有的数据的超大文件。

■注：对于磁带存储媒介，一般不设置 Maximum Volume Bytes，因为我们通常希望磁带在它的容量范围内保存尽可能多的数据。如果一个卷在创建时设置了存储上限，该卷就会一直保持该上限而忽略 Pool 资源对它的上限设置。此时必须使用 Bacula 控制台更新该卷，从而改变它的上限。

在 Pool 的定义中，还设置了“AutoPrune = Yes”，有了该设置，一旦 Volume Retention 保

存周期过期，Bacula 就会把备份信息从任务日志中删除。不过有时候并不希望发生这种情况，例如，对于某些归档备份，希望尽可能久地保存备份数据，而并不想从任务日志中删除这些备份的记录。在这种情况下，为了安全起见，需要把 AutoPrune 设置成 No。

■注：即使把 Pool 设置成 “AutoPrune = No” 和 “Volume Retention = 10 years”，备份数据的保存期限最多也就和磁带的寿命一样长。无论在 Bacula 任务日志上保存了哪些信息，改写备份数据或者备份数据遭到破坏都会造成数据无法恢复。

Volume Retention 告诉 Bacula 当前卷上的备份数据应该保存多久。我们对池中的 Volume Retention 属性设置了不同的值，打算把完全备份卷上的数据保存 6 个月，而把增量备份卷上的数据保存 14 天。当然也可以延长保存期限。当保存期限过期后，Bacula 会认为卷上的数据已经过期，而将这些数据重写。

在 Pool 定义中设置的 “Recycle = Yes” 的含义是，一旦卷上的数据被判定过期了，那么该卷对所有的池重新变得可用，可以再次被写入数据。设置 “Recycle Oldest Volume = Yes” 意味着如果有多个卷可用，那么最先可用的卷将被最先使用。这样做有一个明显的优点就是最大限度地推迟了可用卷上的原有数据被破坏的时间，因此在数据被破坏之前，仍然可以从这些卷上恢复它们。

Pool 资源定义中最后的设置项是 Label 项。Bacula 用 Label 给卷赋予一个名称或标记。在添加一个新磁带的时候，来自自动装载机并带有条形码标记的磁带上的数据是不会被自动写入 Bacula 的，需要手动标记该磁带。在把磁带作为存储媒介的情况下，一小段数据被写到磁带开始的位置，这段数据赋予该卷（磁带）一个标记名。一旦磁带被标记了，Bacula 就根据标记识别出哪个磁带被插入磁带机。我们使用 “Full-” 以及 “Incremental-” 再加上后缀 “\${Year}-\${Month:p/2/0/r}-\${Day:p/2/0/r}” 生成带有标记的卷，如 “Full-2009-01-01”。要获知适用于 Bacula 的可变扩展名的完整信息，请访问网址 http://www.bacula.org/en/rel-manual/Variable_Expansion.html#VarsChapter。

■注：有一个关于备份策略的讨论涉及如何使用 Bacula 管理磁带和备份周期，请访问 http://www.bacula.org/en/rel-manual/Backup_Strategies.html。

console 服务（console service）使 Director 守护进程可以轮询 File 守护进程或 Storage 守护进程，并报告二者的状态信息。Director 守护进程必须提供密码用于验证，还必须确定允许监控的 Bacula 服务的名称。在 Director 守护进程配置文件中，有如下所示的配置。

```
Console {
  Name = example-mon
  Password = "directormonpassword"
  CommandACL = status, .status
}
```

信息服务（messaging service）定义在 Director 配置文件中，它是一种通信方式，此种方式的通信应由 Bacula 发起，并通过 Director 守护进程发送信息，但该通信不能由独立的客户机发起。当某个事故发生或者一个任务结束时，这些信息就会被发送到标准的信息服务，该

信息服务在 Director 配置文件中配置。命名为 Standard 的信息服务资源定义在 Messages 部分，如下所示。

```
Messages {
    Name = Standard
    mailcommand = "/usr/sbin/bsmtp -h mail.example.com -f \"\\(Bacula\\) %r\" -s \"Bacula: ➡
    %t %e of %c %l\" %r"
    operatorcommand = "/usr/sbin/bsmtp -h mail.example.com -f \"\\(Bacula\\) %r\" ➡
    -s \"Bacula: Intervention needed for %j\" %r"
    mail = root@localhost = all, !skipped
    operator = root@localhost = mount
    console = all, !skipped, !saved
    append = "/var/spool/bacula/log" = all, !skipped
}
Messages {
    Name = Daemon
    mailcommand = "/usr/sbin/bsmtp -h mail.example.com -f \"\\(Bacula\\) %r\" ➡
    -s \"Bacula daemon message\" %r"
    mail = root@localhost = all, !skipped
    console = all, !skipped, !saved
    append = "/var/log/bacula.log" = all, !skipped
}
```

Bacula 使用 mailcommand 和 operatorcommand 命令发送表示其状态信息的电子邮件，该信息可以是警告信息，或者任务完成信息。Bacula 使用自己的二进制邮件发送器 bsmtp 把信息发送到合适的邮箱地址。-f 设置邮件的 From 头，-s 设置邮件的 Subject 头。我们需要更改 mail 和 operator 设置以适应当前的环境，做法是把它们的 e-mail 地址设置成 Bacula 服务器管理员的电子邮箱（在我们的例子里，该邮箱是 admin@example.com）。类似于在 mail 定义中设置的“成功/失败”信息，可以给另外一个邮箱地址设置 operator 类型信息，例如“Please mount tape”。还要添加“-h mail.example.com”设置，为 mailcommand 和 operatorcommand 命令指定邮件服务器。

在下一章节，通过所要配置的文件介绍 Storage 守护进程。

2. bacula-sd.conf

Bacula 把 Storage 守护进程的配置文件保存在文件/etc/bacula/bacula-sd.conf 中。Bacula 提供的 Storage 配置文件有各种不同的设备配置实例。用户必须把 Bacula 提供的 Storage 配置文件作为配置向导，以用于配置不同类型的存储设备。

```
Storage {
    Name = example-sd
    SDPort = 9103
    WorkingDirectory = "/var/spool/bacula"
    Pid Directory = "/var/run"
    Maximum Concurrent Jobs = 20
}
Director {
    Name = example-dir
    Password = "storagepassword"
}
```


在文件/etc/bacula/bacula-sd.conf中，设置 Storage 守护进程的 Name 属性为 example-sd。在示例网络中，可能还会有其他 Storage 守护进程，因此 Storage 守护进程的名字必须是唯一的。Director 部分设置了 Bacula Director 守护进程的 Name 属性，并设置 Password 属性为 storagepassword（storagepassword 属性值在/etc/bacula/bacula-dir.conf 文件中也设置过）。在 Device 部分，Name 属性匹配 Director 配置文件中 Storage 定义的 Device 属性。“Media Type = FileStorage”匹配 Director 配置文件和 Storage 配置文件中的 Media Type 属性。

```
Device {
    Name = FileStorage
    Media Type = File
    Archive Device = /data/backups/bacula/FileStorage
    LabelMedia = yes;
    Random Access = Yes;
    AutomaticMount = yes;
    RemovableMedia = no;
    AlwaysOpen = no;
}
```

这里，指定了存储备份的设备。上面的设置没有使用任何磁带机或者自动加载器，因此这里的设置是一种简单的基于磁盘的存储设置。另外，还指定了存储备份文件的地方。在本例子中，我们准备把备份文件存储到/data/backups/bacula/FileStorage 目录下，并保持其他设置项的默认值不变。

■注：可以在“http://www.bacula.org/en/rel-manual/Storage_Daemon_Configuration.html#StoredConfChapter”上查阅到更多关于配置 Storage 守护进程的信息。

```
Director {
    Name = example-mon
    Password = "directormonpassword"
    Monitor = yes
}
Messages {
    Name = Standard
    director = example-dir = all, !skipped, !restored
}
```

bacula-dir.conf 中的 Monitor 或 Console 定义了资源文件“Director { ...Name = example-mon ... }”。该定义用于监控资源，并允许 Director 守护进程获取 Storage 守护进程的状态信息。这要求 Director 配置文件中设置的密码必须和这里设置的密码一样。信息资源允许存储服务器通过 Bacula Director 守护进程发送信息性消息。

3. bacula-fd.conf

在能够执行一次备份任务之前，必须要做的最后一件事情是在 headoffice.example.com 主机上安装 bacula-fd 软件。Red Hat 和 Ubuntu 主机必须安装 bacula-common 和 bacula-client，安装过程中将下载所有的依赖软件。一旦安装好这些软件，就可以编辑/etc/bacula/bacula-fd.conf 文件，给该文件添加以下信息。


```
Director {
    Name = example-dir
    Password = "headofficepassword"
}
FileDaemon {
    Name = headoffice-fd
    FDport = 9102
    WorkingDirectory = /var/spool/bacula
    Pid Directory = /var/run
    Maximum Concurrent Jobs = 20
}
Director {
    Name = example-mon
    Password = "directormonpassword"
    Monitor = yes
}
Messages {
    Name = Standard
    director = example-dir = all, !skipped, !restored
}
```

在 File 守护进程的配置文件脚本中, 声明“Director {}”定义部分用于连接 Bacula Director 守护进程。该声明部分必须含有所要连接的 Director 守护进程的名字, 并含有 Client 资源中为 headoffice-fd 设置的密码。FileDaemon 指定了 Name、FDPort 和 Maximum Concurrent Jobs 的设置。这里的 Name 设置必须和 Client 中的 Name 设置一样, 这里的 FDPort 设置为默认的 TCP 9102 端口。Maximum Concurrent Jobs 规定了同一时间能够连接到该主机的最大任务数。

Director 和 Messages 定义与 Storage 配置文件中的同名定义是一样的。它们用于允许 Director 守护进程获取 File 守护进程的状态信息, 以及指定在发生错误的时候信息被发往哪里。

4. 测试配置文件

现在, 输入以下命令, 可以测试 bacula-dir.conf 配置文件的句法。

```
$ sudo /usr/sbin/bacula-dir -t -c /etc/bacula/bacula-dir.conf
```

对于 File 守护进程和 Storage 守护进程, 输入以下命令。

```
$ sudo /usr/sbin/bacula-fd -t -c /etc/bacula/bacula-fd.conf
```

或者是以下这行命令。

```
$ sudo /usr/sbin/bacula-sd -t -c /etc/bacula/bacula-sd.conf
```

如果命令没有返回错误, 就可以使用 service 或者 invoke-rc.d 命令启动 Bacula 服务。怎么判断服务器正在运行呢? 可以使用 status 参数查看服务是否正在运行。

```
$ sudo /etc/init.d/bacula-dir status
bacula-dir (pid 8774) is running...
$ sudo /etc/init.d/bacula-fd status
bacula-fd (pid 8757) is running...
$ sudo /etc/init.d/bacula-sd status
bacula-sd (pid 8738) is running...
```


5. bconsole.conf

现在，必须安装 Bacula 控制台，用它来管理和监控备份服务。bconsole 的配置文件是/etc/bacula/bconsole.conf。

根据自身的需求，可以安装 GUI 方式的 Bacula 控制台和基于命令行的 Bacula 控制台，还可以安装多个 Bacula 控制台。首先安装基于命令行的 Bacula 控制台，在本章稍后将安装 GUI 方式的 Bacula 控制台。在/etc/bacula/bconsole.conf 文件中，做如下定义。

```
Director {
  Name = example-dir
  DIRport = 9101
  Address = backup.example.com
  Password = "directorpassword"
}
```

该定义含有的配置项（Name、port、Address 和 Password）和 bacula-dir.conf 配置文件中的 Directory 含有的配置项一样。

13.5.3 使用 bconsole 管理 Bacula

在配置好 Bacula 之后，可以通过命令 service 或者 invoke-rc.d 启动 Bacula 服务，命令的选择取决于操作系统。在 Red Hat 主机上，我们输入以下命令启动 Bacula 服务。

```
$ sudo /sbin/service bacula-dir start
$ sudo /sbin/service bacula-sd start
$ sudo /sbin/service bacula-fd start
```

而在 Ubuntu 主机上，输入以下命令。

```
$ sudo /usr/sbin/invoke-rc.d bacula-dir start
$ sudo /usr/sbin/invoke-rc.d bacula-sd start
$ sudo /usr/sbin/invoke-rc.d bacula-fd start
```

现在，可以启动 Bacula 控制台程序了，使用的 bconsole 命令如下所示。

```
$ sudo /usr/sbin/bconsole
```

首先，输入“status all”命令，列出 Director 守护进程的当前状态。该命令给出了 Director 守护进程运行状态的详细信息，还包括 Storage 守护进程和 File 守护进程的状态信息。

为了知道 Director 守护进程本身正在做什么，可使用“stat dir”命令。该命令给出挂起任务、当前运行任务和已完成任务的概况。输入以下命令访问 Bacula 控制台。

```
$ sudo /usr/sbin/bconsole
Connecting to Director 127.0.0.1:9101
1000 OK: example-dir Version: 2.4.2 (26 July 2008)
Enter a period to cancel a command.
*stat dir
example-dir Version: 2.4.2 (26 July 2008) i386-redhat-linux-gnu redhat
Daemon started 04-Mar-09 07:59, 0 Jobs run since started.
  Heap: bytes=27,197 max_bytes=27,197 bufs=184 max_bufs=184
```



```
Scheduled Jobs:
Level      Type      Pri  Scheduled      Name      Volume
=====
Incremental Backup  10   04-Mar-09 23:05  headoffice.example.com  *unknown*
Full       Backup   11   04-Mar-09 23:10  BackupCatalog            *unknown*
=====

Running Jobs:
No Jobs running.
=====

No Terminated Jobs.
=====
*
```

正如我们所看到的，目前没有任务完成，也没有任务正在运行。有两个任务安排在 3 月 4 号 23: 05 执行，其中一个任务是备份 headoffice.example.com 主机，另一个是备份任务日志。下面多花点时间探讨一下 bconsole。例如，敲入 hand，按回车，就会看到命令行下可用的命令列表。表 13-3 给出了最有用的一些命令。

表 13-3 有用的 Bacula bconsole 命令

参数	描述
Run	开始一个任务（备份任务）
Cancel	取消当前任务
Mess	显示任何当前信息
Restore	管理恢复任务
Label	给一个磁带或一个媒介卷（media volume）加上标签
update volume	允许更新任何特定卷的属性
stat dir	获取 Director 守护进程的当前状态
stat client	获取 File 守护进程的当前状态
stat storage	获取 Storage 守护进程的当前状态
Mount	挂载一个媒介卷（例如，把一个媒介卷载入到一台磁带机中）
Umount	卸载一个媒介卷（例如，从磁带机上卸载一个磁带）

■注：另一个关于可用命令的资料在 Bacula 文档站点 “http://www.bacula.org/en/devmanual/Brief_Tutorial.html” 上。

有两种方法执行一个任务或备份。一种方法是使用 run 命令并回车，然后，以一种简单易懂的菜单驱动方式提交备份任务。另一种方法是直接在命令行输入以下命令。

```
* run job=headoffice.example.com level=full priority=7 yes
```

执行 “* stat dir” 命令，就可以看到任务正在运行。

```
Running Jobs:
JobId Level Name      Status
=====
2 Full headoffice.example.com.2009-03-04_08.33.11 is running
=====
```


创建一个新卷，并把备份数据恢复到该卷上。如果在命令行下输入 `mess`，将收到与该任务有关的所有信息。

现在，如果想要终止该任务，可以使用 `cancel` 命令，把 `JobId 2` 作为 `cancel` 命令的一个参数（可以从先前的 `Running Jobs` 输出找到需要用到的 `JobId`）。

```
*cancel jobid=2
```

`cancel` 命令可以不带参数执行，它会为我们自动选择 `JobId`（在当前只有一个任务正在运行时，否则，它给出一个任务列表供我们选择）。再次运行该任务，这次不取消它，直到任务完成。在 `Terminated Jobs` 部分做一个 `stat dir` 操作，得到如下信息。

JobId	Level	Files	Bytes	Status	Finished	Name
4	Full	1,198	12.60 M	OK	04-Mar-09 08:53	headoffice.example.com

上述输出表明，我们对主机 `headoffice.example.com` 上的 `Linux FileSet` 所指定的文件集合做了一次完全备份。可以看到备份拥有的文件数目和备份数据的大小。接下来用 `restore` 命令恢复 `/etc/hosts` 文件。

```
* restore
To select the JobIds, you have the following choices:
  1: List last 20 Jobs run
  2: List Jobs where a given File is saved
  3: Enter list of comma separated JobIds to select
```

这里，列表中的可用选项实际上有 12 项，不过我们只用选项 2 找出备份任务的 `JobId`（根据先前的 `Terminated Jobs` 输出，可以知道该 `JobId` 是 4）。

```
Select item: (1-12): 2
Automatically selected Client: headoffice-fd
Enter Filename (no path):hosts
+-----+-----+-----+-----+-----+-----+-----+
| JobId | Name       | StartTime           | JobType | JobStatus | JobFiles | JobBytes |
+-----+-----+-----+-----+-----+-----+-----+
| 4     | /etc/hosts | 2009-03-04 08:52:55 | B       | T         | 1198     | 12606355 |
+-----+-----+-----+-----+-----+-----+-----+
```

现在，知道了 `JobId`，接着用选项 3 恢复主机上的文件，如下所示。

```
Select item: (1-12): 3
Enter JobId(s), comma separated, to restore: 4
You have selected the following JobId: 4
Building directory tree for JobId 4 ... ++++++
1 Job, 1,174 files inserted into the tree.
cwd is: /
$ cd /etc
cwd is: /etc/
$ mark hosts
1 file marked.
$ done
Bootstrap records written to /var/spool/bacula/example-dir.restore.1.bsr

The job will require the following
      Volume(s)           Storage(s)           SD Device(s)
```



```
=====
Full-2009-03-04      File      FileStorage

1 file selected to be restored.

Using Catalog "MyCatalog"
Run Restore job
JobName:      RestoreFiles
Bootstrap:    /var/spool/bacula/example-dir.restore.1.bsr
Where:        /tmp/bacula-restores
Replace:      always
FileSet:      Linux
Client:       headoffice-fd
Storage:      File
When:         2009-03-04 09:07:47
Catalog:      MyCatalog
Priority:      10
OK to run? (yes/mod/no):yes
```

在选择要恢复的文件时，可以使用 `ls` 命令和 `cd` 命令列出目录内容，并更改当前目录以搜索这些文件。

```
cwd is: /
$ cd /etc
cwd is: /etc/
$ mark hosts
1 file marked.
$ done
```

在使用 `cd` 命令浏览备份文件集合后，使用 `mark` 命令选中要恢复的文件。为了递归地选中同一目录下的所有内容，可以使用“`mark *`”命令。当选择好所要恢复的文件时，使用 `done` 命令。

这样得到了执行恢复操作所必需的媒介卷列表。如果这些卷在备份主机上还不可用，则必须载入它们。然后，得到所要执行的恢复任务的概要。记录 `Where` 的内容，即 `/tmp/bacula-restores`，这就是恢复后的文件所在的位置。

观察 `stat dir` 部分和 `Terminated Jobs` 部分，一个已完成的恢复任务将如下所示。

JobId	Level	Files	Bytes	Status	Finished	Name
=====						
	5	1	274	OK	04-Mar-09 09:14	RestoreFiles

现在，可以检查该文件是否已经被恢复到 `/tmp/bacula-restores` 目录下。

```
$ sudo ls -l /tmp/bacula-restores/etc/
total 4
-rw-r--r-- 1 root root 274 2009-03-04 06:54 hosts
```

通常，在确定要执行一个恢复任务之前，通过更改 `Client` 和 `Where` 选项，就可以使用 `restore` 命令把文件恢复到不同的主机和不同的目的地。

13.5.4 使用 Bacula 备份数据库

接下来要向读者展示一种更为高级的应用。在主机 `headoffice.example.com` 上，有一个

MySQL 数据库服务器正在运行。该 MySQL 数据库服务器当前建有 Bacula 任务日志数据库，它同时也包含有我们在第 12 章所创建的 KnowledgeTree DMS 数据库，还包含有其他数据库。我们准备拷贝该 MySQL 服务器上的所有数据库，对该数据库服务器做一个完全备份。为达到上述目的，需要执行 `mysqldump` 命令。但是怎样在凌晨四点的时候执行该备份，还有怎样才知道备份已经结束并得到数据副本呢？我们准备使用一个脚本执行该备份任务，并把备份好的文件存储到磁盘上，而且使用 Bacula 管理该脚本的执行，监控该脚本是否成功执行完毕，并再次执行该脚本管理对备份数据库的删除操作。

■注：要确保主机拥有足够的空间存储数据库备份，因为如果耗尽了主机的存储空间就会有麻烦。一种好的做法是把文件备份到一个独立的存储空间，那么一旦耗尽了存储空间，也不会影响到主机上的其他进程。

Bacula 在 Job 资源上提供两个操作指令，现在可以用到它们了，即 `Client Run Before Job` 和 `Client Run After Job`。正如它们名字所表示的那样，这两个指令可用来执行目标主机上的脚本。`Client Run Before Job` 将在目标主机上执行一个命令或脚本，而且要求在继续执行后续任务之前该指令所执行的命令或脚本必须成功完成。如果该指令没有执行成功，任务将会中止并产生一个错误信息。`Client Run After Job` 同样要求在目标主机上的指定脚本或命令必须成功执行完毕，否则，该指令将终止任务的进一步执行并提交一个错误信息。回顾先前定义的主机 `headoffice.example.com` 的 Job 资源，然后对该资源做如下修改。

```
Job {
  Name = headoffice.example.com
  Client = headoffice-fd
  Enabled = yes
  JobDefs = "DefaultLinux"
  Client Run Before Job = "/usr/local/bin/mysql_backup start"
  Client Run After Job = "/usr/local/bin/mysql_backup stop"
}
```

在目标主机的 `/usr/local/bin` 目录下必须有 `mysql_backup` 脚本存在。现在要做两件事情，首先要做的事情是在备份任务启动之前，使用 `start` 参数执行该脚本。该脚本执行 `mysqldump` 命令，开始备份 MySQL 数据库。如果脚本成功执行完毕，则接下来要做的事情是按照 `FileSet` 中的指令对系统其余文件进行备份。如果这些备份也成功完成，Bacula 将执行 `mysql_backup stop` 指令，把数据库备份文件删除。在此，需要亲自创建 `mysql_backup` 脚本，表 13-3 给出一个很好的脚本模板。

列表 13-3 MySQL 备份脚本

```
#!/bin/bash

USER='root'
PASSWD='mysqlrootpassword'

case $1 in
  start)
```



```
mysqldump --opt --all-databases -u $USER -p$PASSWD ➡
> /var/lib/mysql/backups/mysql.sql
    if [ $? -eq 0 ] ; then
        echo "backup successful"
    else
        echo "backup failed"
        exit 1;
    fi

;;
stop)
if [ -e /var/lib/mysql/backups/mysql.sql ] ; then
    rm -f /var/lib/mysql/backups/mysql.sql
    if [ $? -eq 0 ] ; then
        echo "removal of file successful"
    else
        echo "failed to remove file"
        exit 1;
    fi

fi
;;
esac
exit 0
```

■注：该脚本含有敏感的密码信息，建议读者按照以下权限设置谨慎地保存该脚本，即 `chmod 500`。

在目标主机 `headoffice.example.com` 上，必须保证 `/var/lib/mysql/backups` 目录已被创建，并且该目录有足够的空间保存 MySQL 数据库的备份文件。

现在，用以下命令在 `bconsole` 中执行备份任务。

```
> run job=headoffice.example.com client=headoffice-fd yes
```

我们可以用 “`stat client=headoffice-fd`” 命令显示目标主机上备份任务的执行情况，也可以用 “`stat dir`” 命令查看 Bacula 服务器上的任务进度。而且，可以通过 `mess` 命令或者直接打开 Bacula 日志来查看所有的任务信息。可以看到有个已完成的任務，它的结束代码是 “Backup OK”。如下所示，Client Run Before Job 指令和 Client Run After Job 指令都成功执行。

```
05-Mar 07:21 example-dir: Start Backup JobId 12, Job=headoffice.example.com. ➡
2009-03-05_07.21.41
05-Mar 07:21 example-dir: Recycled volume "Full-2009-03-04"
05-Mar 07:21 headoffice-fd: ClientRunBeforeJob: run command ➡
"/usr/local/bin/mysql_backup start"
05-Mar 07:21 headoffice-fd: ClientRunBeforeJob: backup successful
05-Mar 07:21 example-sd: Labeled new Volume "Full-2009-03-04" on device ➡
"FileStorage" (/data/backups/bacula/FileStorage).
05-Mar 07:21 example-sd: Wrote label to prelabeled Volume "Full-2009-03-04" on ➡
device "FileStorage" (/data/backups/bacula/FileStorage)
05-Mar 07:22 headoffice-fd: ClientAfterJob: run command "/usr/local/bin/mysql_backup ➡
stop"
```



```

05-Mar 07:22 example-sd: Job write elapsed time = 00:00:49, Transfer rate = 262.7 K→
bytes/second
05-Mar 07:22 headoffice-fd: ClientAfterJob: removal of file successful
05-Mar 07:22 example-dir: Bacula 2.0.3 (06Mar07): 05-Mar-2009 07:22:33
JobId:                12
Job:                  headoffice.example.com.2009-03-05_07.21.41
Backup Level:         Full
Client:               "headoffice-fd" 2.0.3 (06Mar07) i386-redhat-linux-gnu, →
redhat,
FileSet:              "Linux" 2009-03-04 08:50:13
Pool:                 "Full" (From Job FullPool override)
Storage:              "File" (From Pool resource)
Scheduled time:       05-Mar-2009 07:21:25
Start time:           05-Mar-2009 07:21:44
End time:             05-Mar-2009 07:22:33
Elapsed time:         49 secs
Priority:              10
FD Files Written:     1,200
SD Files Written:     1,200
FD Bytes Written:     12,738,605 (12.73 MB)
SD Bytes Written:     12,872,793 (12.87 MB)
Rate:                 260.0 KB/s
Software Compression: 85.6 %
VSS:                  no
Encryption:           no
Volume name(s):       Full-2009-03-04
Volume Session Id:    4
Volume Session Time:  1236255409
Last Volume Bytes:    12,928,002 (12.92 MB)
Non-fatal FD errors:  0
SD Errors:            0
FD termination status: OK
SD termination status: OK
Termination:          Backup OK

```

现在，我们知道了如何在 Bacula 备份服务器上管理复杂的备份，学会了用 Bacula 备份一个复杂的文件系统，或者备份数据目录。如果只把数据目录作为备份目标，那么就必须考虑在系统出错的时候如何恢复数据。

接下来，介绍使用 Bat 控制台管理 Bacula 配置。

13.5.5 介绍 Bat 控制台

使用基于文本的控制台安装和管理 Bacula 服务是一种便捷而简单的方法。但是有的人更喜欢 GUI 图形界面，因此 Bacula 创建了一款叫做 Bat 的、优秀的、简洁的图形界面程序用于管理其 Bacula Director 守护进程。

在 Red Hat 上，用于安装 Bat 程序的软件包是 bacula-bat，它可以在 <http://sourceforge.net> 站点获得；而在 Ubuntu 上，Bat 的安装包是 bacula-console-qt，它可以从 Ubuntu 的在线资源库获得。在安装 Bat 时，还需要安装依赖软件包 qt4。

Bat 的配置与 bconsole.conf 中的配置十分类似，它保存在/etc/bacula/bat.conf 文件中。

```
Director {
  Name = example-dir
  DIRport = 9101
  address = backup.example.com
  Password = "directorpassword"
}
```

可以看到，上述定义包含 Bat 要连接的 Bacula Director 守护进程的名字，以及访问 Director 守护进程必需的地址、端口和密码。可以用 Bat 管理多个 Director 守护进程，也可以往上述定义添加 Director 配置文件的其他内容。

从 Linux 桌面菜单启动 Bat，如图 13-3 所示。



图 13-3 启动 Bat

这里需要输入 root 用户密码以获得管理员权限。然后 Bat 将启动，展现如图 13-4 所示的界面。

图 13-4 展示出一个简洁、友好的界面。左边是导航菜单，右边是信息窗口。视图底部有一个命令文本框，它允许我们直接给控制台输入命令，并在信息窗口中查看命令执行结果。该命令文本框接受所有标准 Bacula 命令。

在左边的文本菜单栏，可以看到“Clients”、“FileSets”和“Jobs”操作选项。在图 13-5 中，观察“Clients”信息窗口，可以看到“headoffice-fd”客户机被列举出来，一同列出的详细信息包括“Client Name”（客户机名）、“File Retention”（备份文件保存周期）和“Job Retention policies”（任务周期性策略）等。

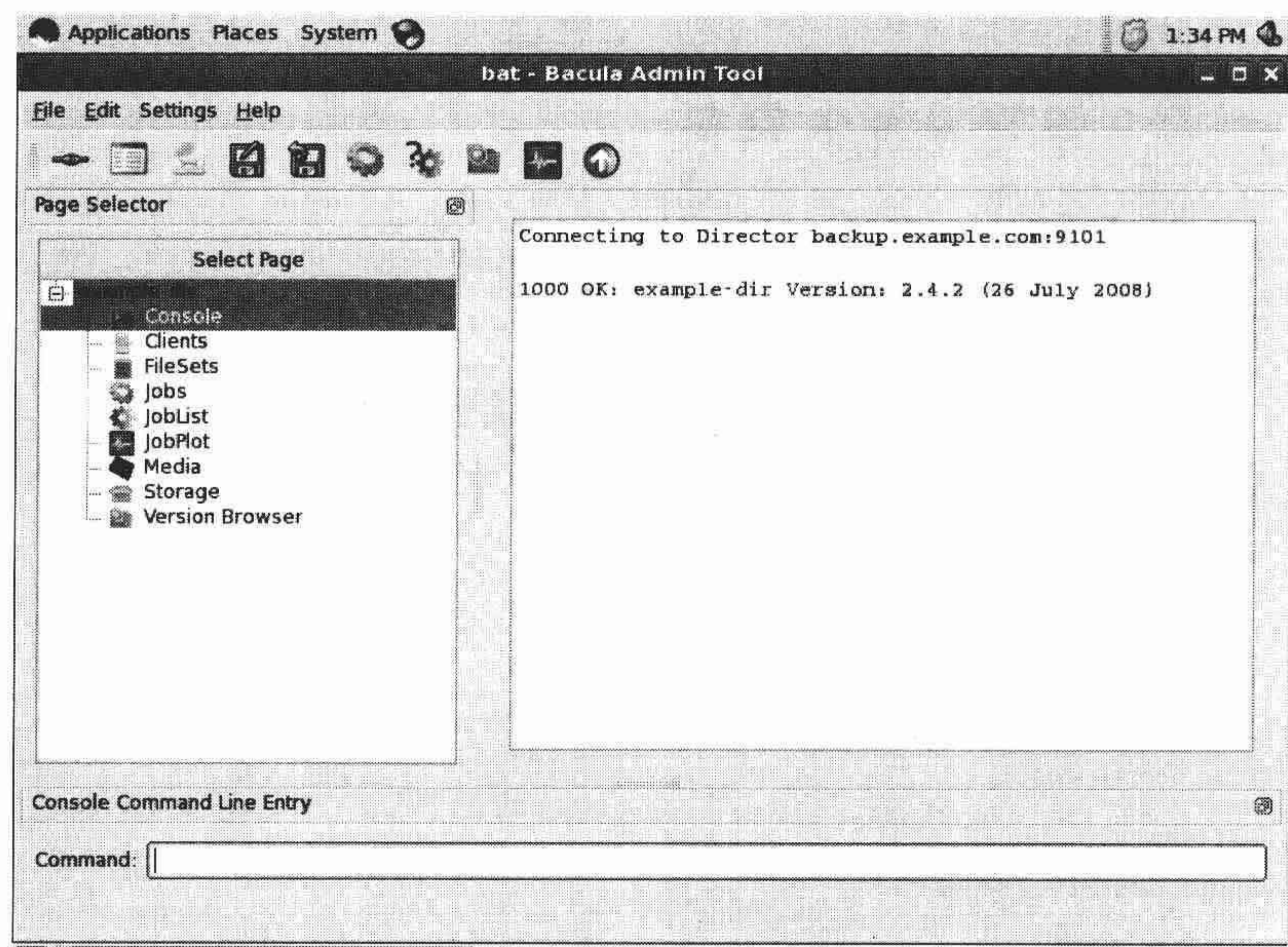


图 13-4 Bat 控制台

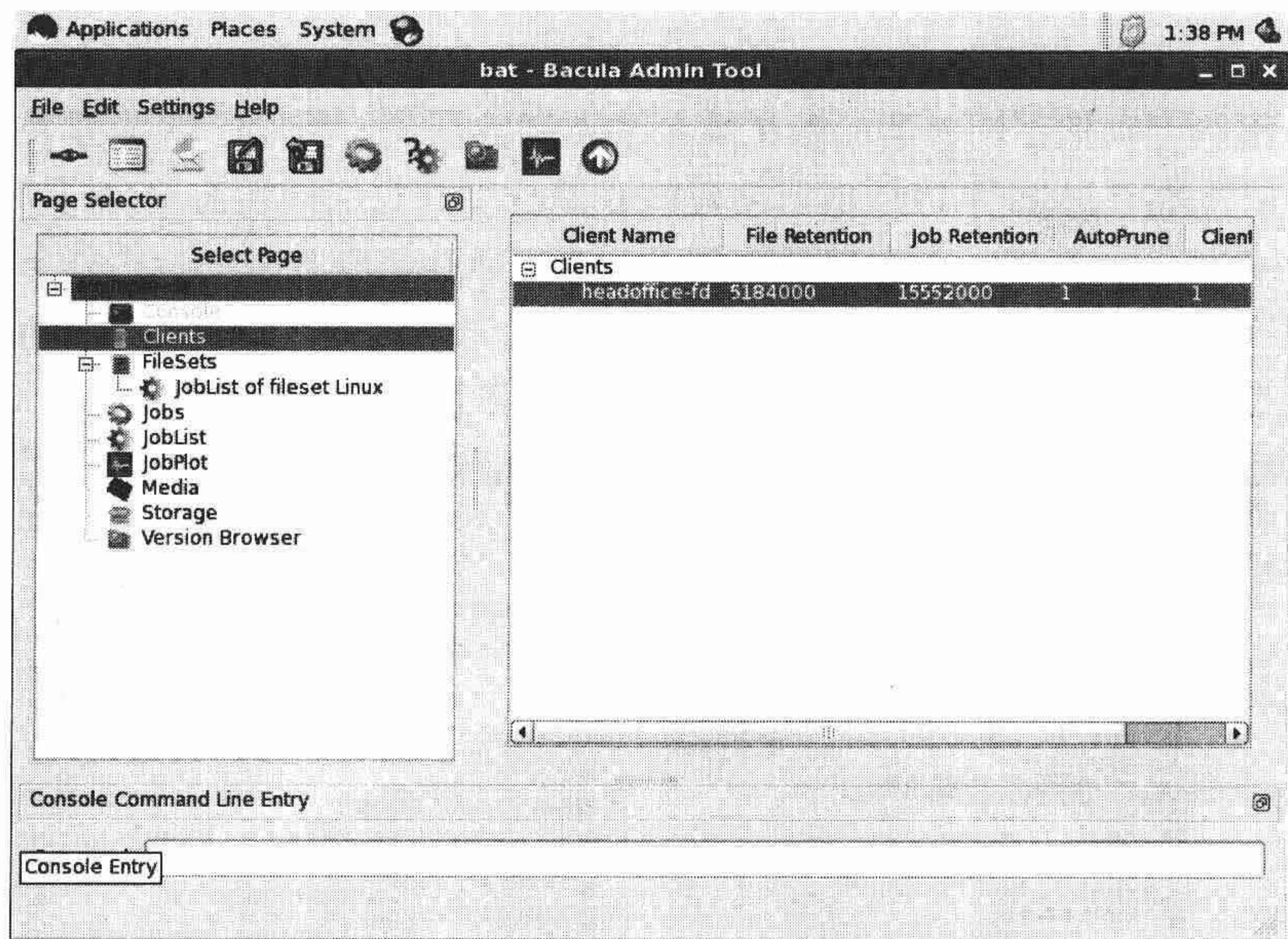


图 13-5 客户机列表

可以用鼠标与 Bat 控制台交互，选中一项资源（resource）单击鼠标右键，将会弹出一列菜单选项供用户选择，如图 13-6 所示。

可以从图 13-6 中看到，在和 Clients 资源（Clients resource）交互的时候，可以选择多个选项。在本例中，选择清除（Purge）与客户机 headoffice-fd 相关的所有任务。这意味着将从数据库中删除与该客户机相关的所有记录（但并不会真正从存储媒介上删除这些信息）。

图 13-7 显示，鼠标右击 Jobs 资源中的某个任务，可以从 Bat 控制台启动一个备份任务。

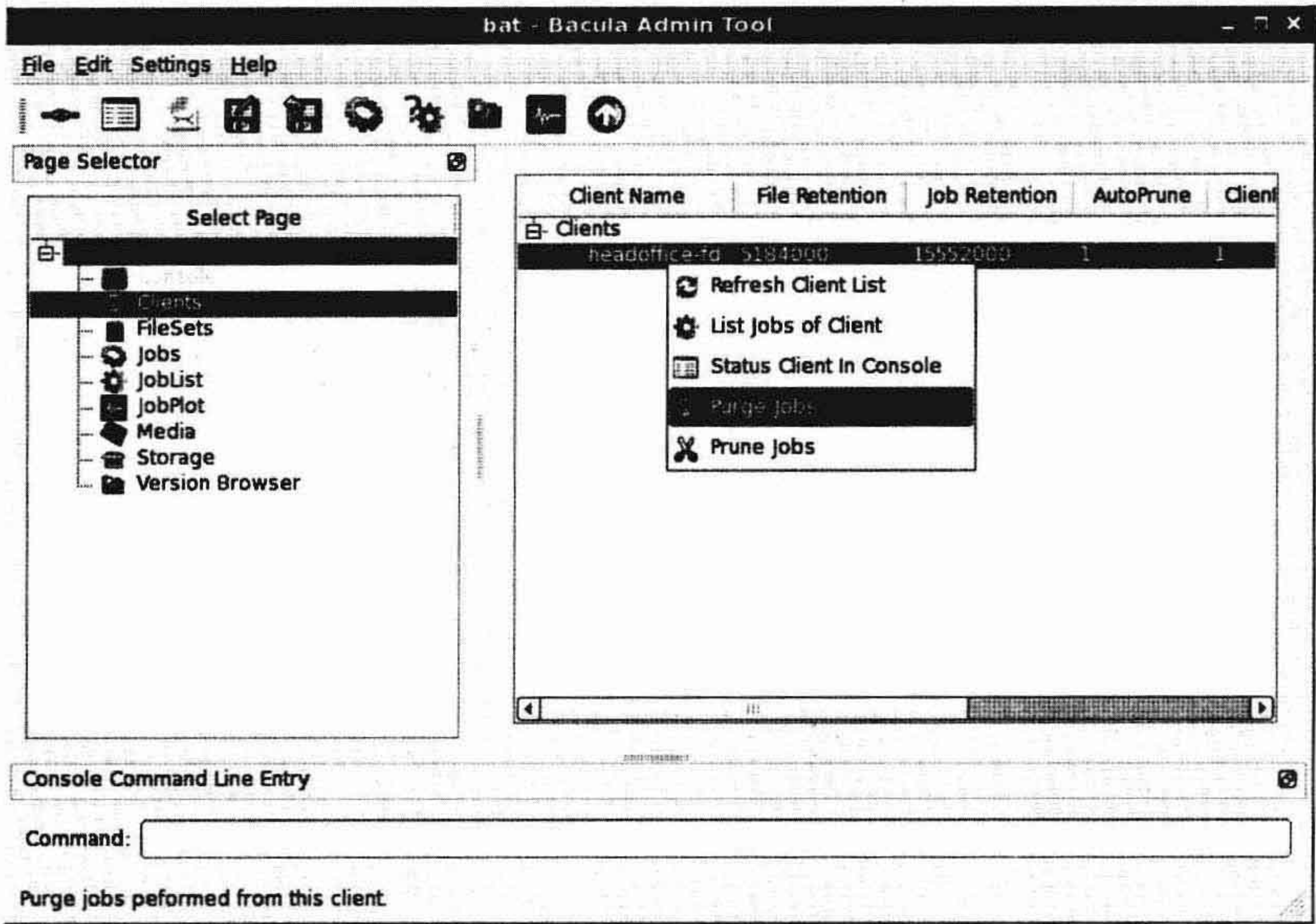


图 13-6 与 Bat 控制台交互

可以从 JobPlot 窗口中看到一个由已提交的任务构成的数据图表。该图表有利于观察备份任务的动向。我们可以把不同时期的数据绘制成不同的图表，借助于这些图表可以动态衡量 Bacula 服务器的性能。Media 窗口列出已创建的存储池以及它们的状态信息(如图 13-8 所示)。

正如图 13-8 所示，其中一个存储卷被列举出来。该存储卷被最后一个任务写入信息，其当前状态为“Full”。这里有多多个操作选项可供选择。如果选择“Delete Volume”，将把该卷从所属池中删掉，并将数据库中与该卷相关的所有记录删除。

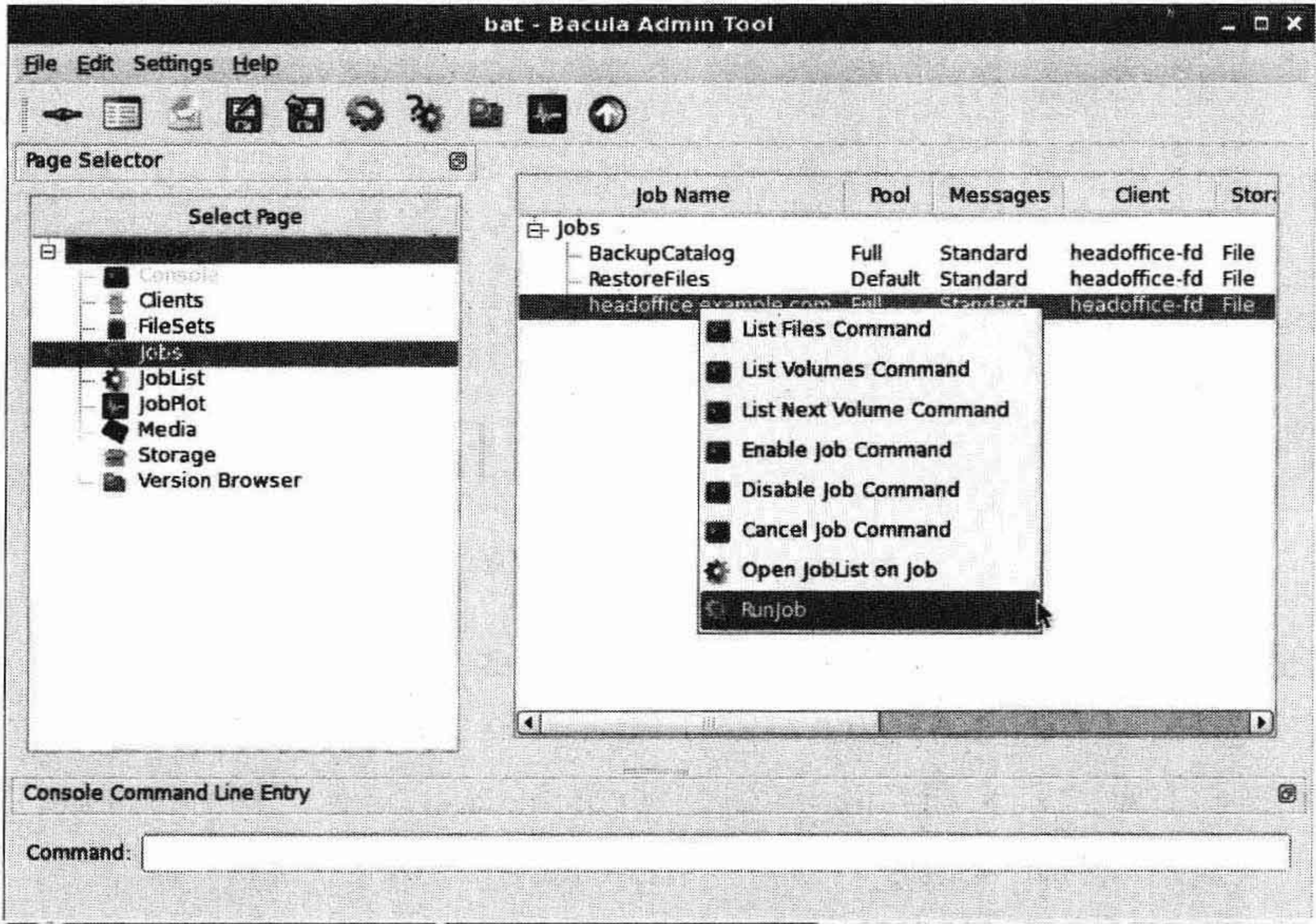


图 13-7 执行一个任务

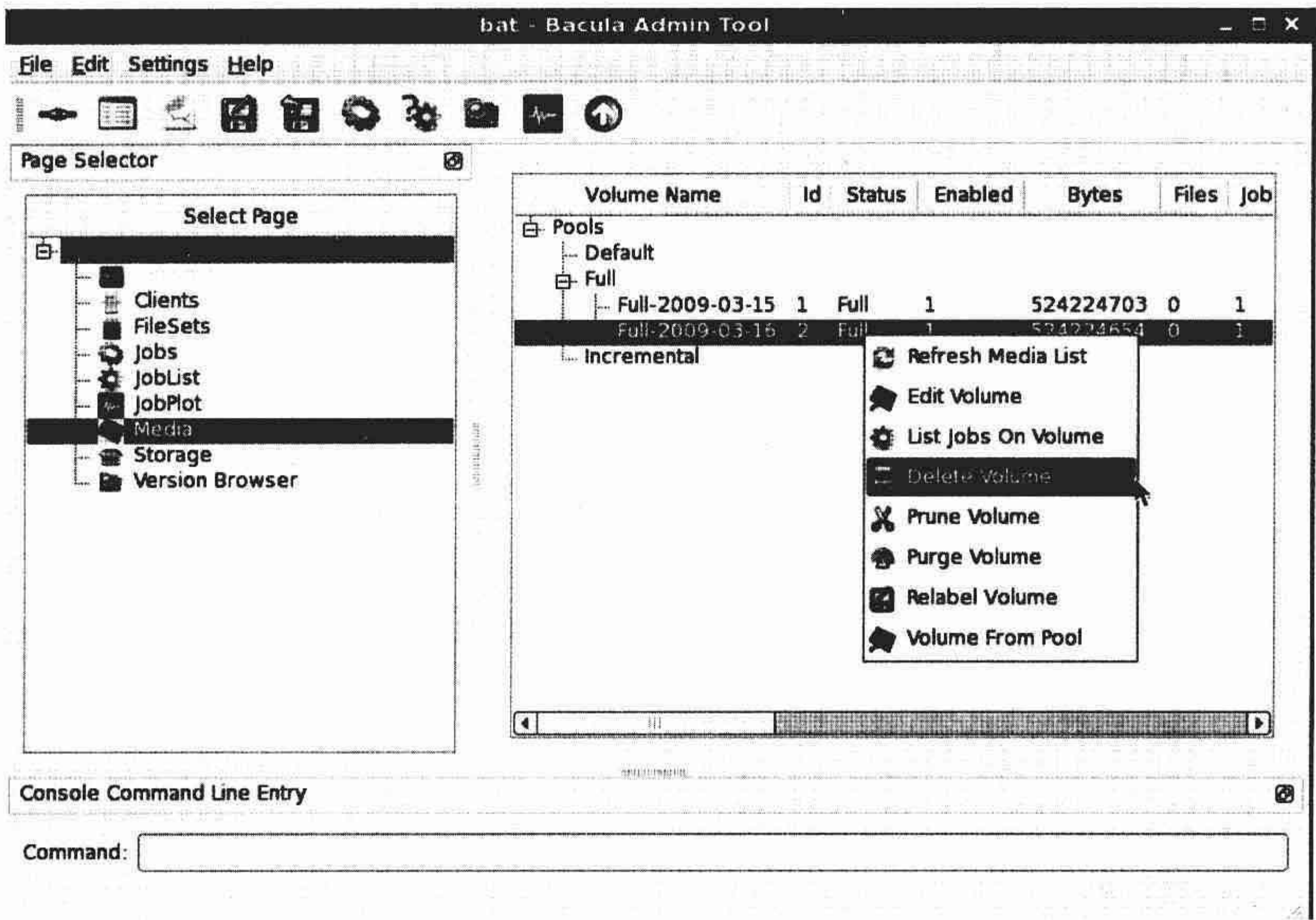


图 13-8 在 Media 窗口中显示存储池

最后，可以查看 Storage 资源的详细信息，如图 13-9 所示。再次对列表某项单击鼠标右键，可以从右键菜单中选择某项菜单进行操作。

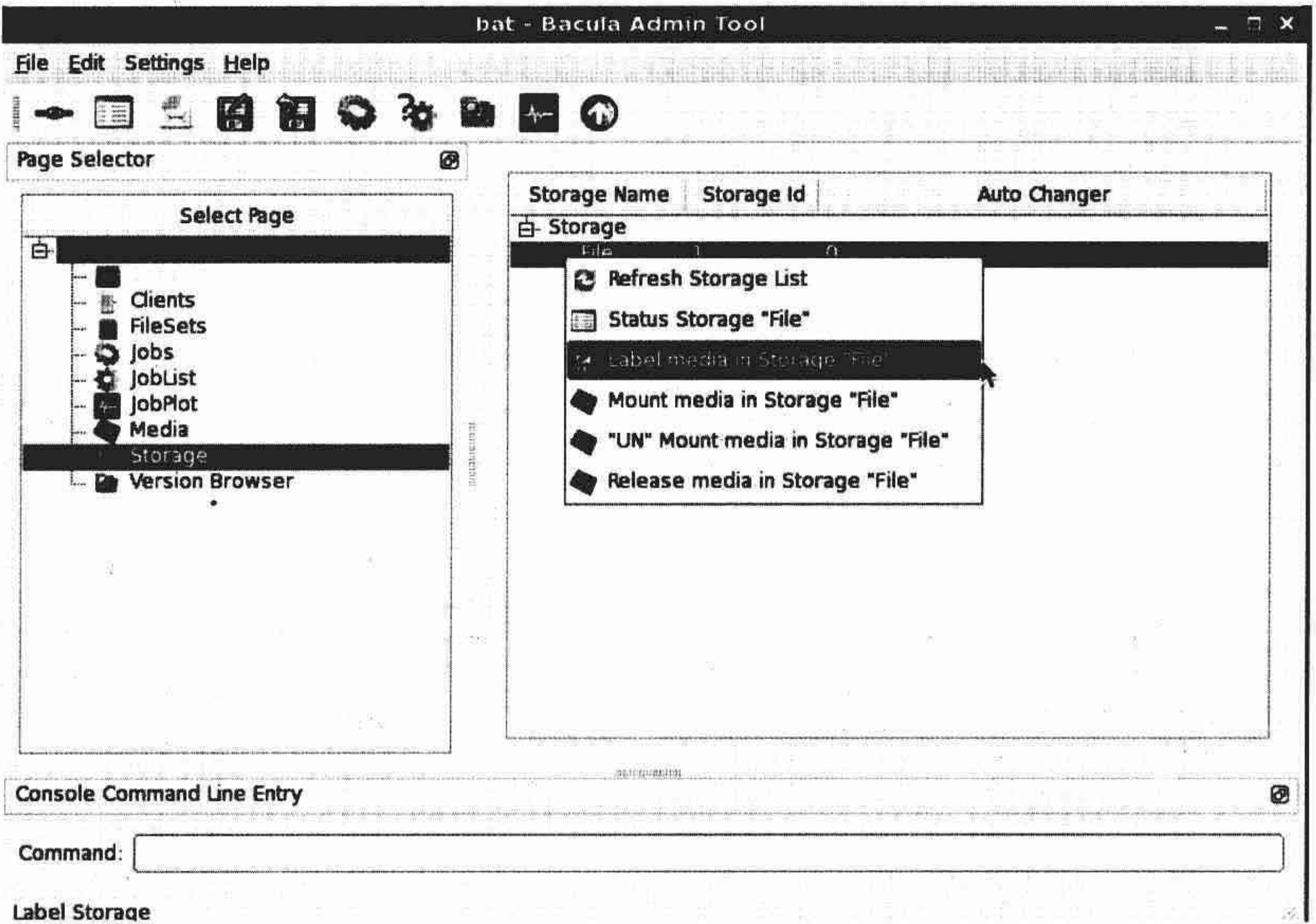


图 13-9 存储媒介选项

通过右键菜单，可以挂载或卸载存储卷，也可以查看存储卷的状态信息。
正如我们所看到的，Bat 控制台是一个功能齐全、操作直观的工具，它可以帮助用户进行 Bacula 服务器的配置管理。

■注：第 19 章将探讨配置管理。在第 19 章，我们就使用一个叫做 Cobbler 的安装管理工具和一个叫做 Puppet 的配置管理工具来就恢复主机数据展开讨论。这两个工具可以结合 Bacula 服务器一起使用，提供一种简单有效的恢复过程。

13.6 小 结

本章探讨了给复杂环境下的单个 Linux 主机或成百上千个服务器及工作站做备份的有效途径，向读者介绍了用于管理备份的软件——rsync 和 Bacula。读者可以结合使用它们，以安全地备份远程主机上的数据。

稍后，在第 19 章中我们将介绍自动化安装系统，结合一种备份方式使用该系统可以快速而高效地恢复数据。

本章讨论了下列主题。

- 在制定灾难恢复计划（disaster recovery planning, DRP）和业务持续管理（business continuity management, BCM）策略的时候，需要考虑到哪些问题。
- 如何使用 rsync 和 SSH，把多个远程主机上的数据备份到一个中心主机上。
- 如何编写一个简单的 Bash 脚本，定期地同步（rsync）远程主机上的数据。
- 如何安装和配置 Bacula 服务器。
- 如何使用 Bacula 服务器执行备份任务。
- 如何使用 Bacula 服务器恢复文件。
- 如何使用 Bacula 服务器对 MySQL 数据库做高级备份。
- Bat 控制台，Bacula 服务器的高级 GUI 图形界面。

下一章介绍如何管理自己的 VPN 链路和互联网络的安全。

第 14 章 构建 VPN 网络

在前面的几章里讨论了很多服务（比如 e-mail 服务和 Web 服务），这些服务也许已经应用到读者企业或组织机构中。我们介绍了多种方法把这些服务提供给各种用户。这些用户包括因特网用户、移动用户以及其他地方的用户。虽然有些服务部署在局域网内是简单而安全的（比如文件共享和打印共享服务）。但是，如果用户不在局域网，那么就需要一些方法来与它们建立连接，使得他们就像本地用户一样。下面，让我们走近虚拟专用网（virtual private network, VPN）。

VPN 其实是一个运行在公网之上的专有网络，它通常称作“通道”。在 Internet 这样的公共网络上，经常需要保证私人通信的安全，而 VPN 正是用来保护通信的安全的。可以建立从网络设备到网络设备的 VPN，也可以建立从主机到主机的 VPN。举个例子，我们可以在两个公司网络之间架设 VPN，也可以在一个客户端（可以是台式机或者是笔记本）和一个公司网络之间架设 VPN。VPN 上的通信通常需要经过加密和认证，其加密和认证机制包括 SSL 证书、密码或者像令牌和智能卡这样的双重身份认证方法。

本章将介绍如何安装相应的软件（这里所安装的软件是一个叫做 OpenVPN 的开源 VPN 工具）来创建一个 VPN，以及如何配置和生成 VPN 通道。

14.1 示例网络

本章将描述多种 VPN 连接，并且使用在第 6 章所创建的网络作为要配置的示例网络环境。图 14-1 再次显示了该网络。

在当前的示例网络中，有一个总公司网络。该网络有两台中心主机。

- gateway.example.com: 防御主机。它有一个外部的 IP 地址 10.0.2.155 和一个内部的 IP 地址 192.168.0.254。
- headoffice.example.com: 总公司的服务器主机。它有一个内部的 IP 地址 192.168.0.1 和几个穿透网关主机的外部连接。

■注：第 9 章介绍了如何创建 DNS 别名记录服务（比如网关和总公司网络）。

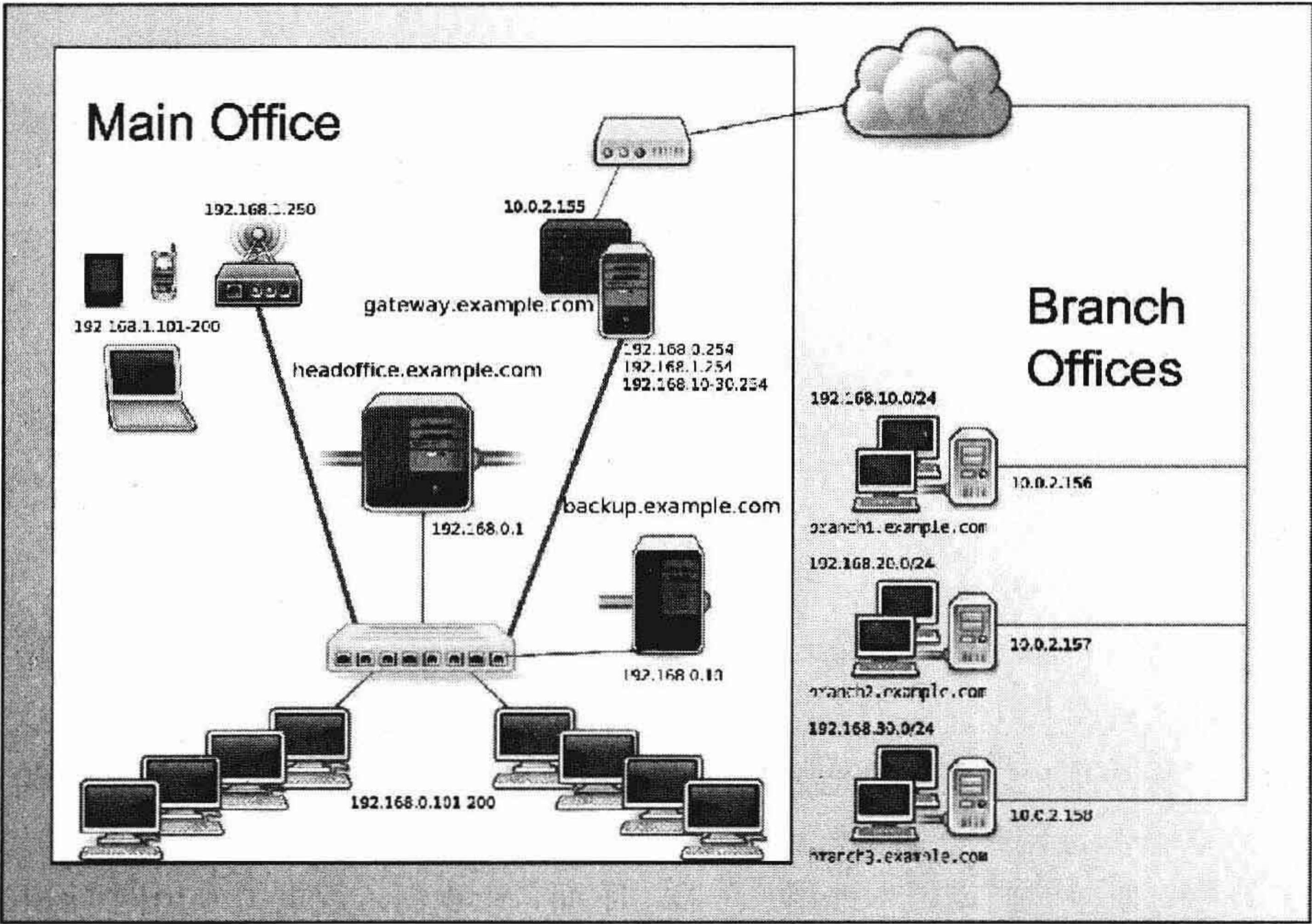


图 14-1 示例网络

还有 3 个分支公司网络，每个网络都有自己的外部 IP 地址和内部 IP 地址域。

- **branch1.example.com**: 一个子公司网络主机，带有外部的 IP 地址 10.0.2.156。该子网络有一个内部的 IP 地址域 192.168.10.0/24。
- **branch2.example.com**: 一个子公司网络主机，带有外部的 IP 地址 10.0.2.157。该子网络有一个内部的 IP 地址域 192.168.20.0/24。
- **branch3.example.com**: 一个子公司网络主机，带有外部的 IP 地址 10.0.2.158。该子网络有一个内部的 IP 地址域 192.168.30.0/24。

14.2 OpenVPN 介绍

OpenVPN (<http://openvpn.net/>) 是一个开源的 SSL VPN 应用程序，它是 James Yonan 编写的，其有效许可证是 GNU GPL。OpenVPN 采用客户端/服务器模式，由一个服务器和多个连接到服务器并创建 VPN 通道的客户端组成。

■注：其他基于 Linux 的 VPN 解决方案也是可取的，包括一些实现了 IPsec 的系统，比如 FreeS/WAN。读者可以在“<http://linas.org/linux/vpn.html>”找到其他可选的 VPN，这是一个非常完整的列表。

OpenVPN 可以运行在多个平台上，包括 Linux、Solaris、Mac OS X 以及 Microsoft Windows。它允许我们把多种客户端连接到 Linux 主机，比如，可以从 Microsoft Windows

的台式机或便携式电脑上连接 VPN 通道。OpenVPN 甚至可以在装有 Windows Mobile 或者 PocketPC 的移动设备上运行，也可以在苹果公司的 iPhone 这样的设备上运行。总之，可以使用 OpenVPN 在各种各样的设备上创建 VPN 通道，从而安全地访问内部网络中的资源。

下一节用多个配置文件描述如何安装和建立 OpenVPN。

14.2.1 安装 OpenVPN

我们需要把 OpenVPN 安装到 VPN 连接的两端。对于主机来说，这意味着把 OpenVPN 服务器安装在两个网络端。如果连接的某一端是支持 OpenVPN 的网络设备，那么只需要把 OpenVPN 服务器安装在作为通道端点的主机一端即可。

首先，把 OpenVPN 服务器安装在防御主机 gateway.example.com 上。该主机在示例网络中是总公司网络的网关，它有一个内部的 IP 地址 192.168.0.254 和一个外部的 IP 地址 192.0.2.155。

OpenVPN 在 Red Hat 和 Ubuntu 主机上都能运行，并用一个软件包安装。在 Red Hat 上，将要安装的 OpenVPN 软件包来自 Dag Wieers 的软件包库（<http://dag.wieers.com/rpm/packages/openvpn/>）。在编写本章时，OpenVPN 的最新版本是 2.09，后面将安装这一版本的软件包。另外，安装 OpenVPN 还需要一个前提条件，那就是数据压缩库 lzo2。

```
$ wget http://dag.wieers.com/rpm/packages/openvpn/openvpn-2.0.9-1.el5.rf.i386.rpm
$ wget http://dag.wieers.com/rpm/packages/lzo2/lzo2-2.02-3.el5.rf.i386.rpm
$ sudo rpm -Uvh lzo2-2.02-3.el5.rf.i386.rpm openvpn-2.0.9-1.el5.rf.i386.rpm
Preparing... ##### [100%]
 1:lzo2 ##### [ 50%]
 2:openvpn ##### [100%]
```

在 Ubuntu 上，安装 openvpn 软件包，并且通常要先安装一些附加的软件包。

```
$ sudo aptitude install openvpn
```

■注：本章所有的例子都使用 OpenVPN2.x 或者更高版本，如果这些例子使用 OpenVPN1.x 发布版可能不会成功。如果读者的系统中没有安装 OpenVPN2.x 或者更高的发布版本，那么需要找一个合适的版本进行安装。

14.2.2 启动和停止 OpenVPN

OpenVPN 在系统中是作为一个服务来运行的。Ubuntu 和 Red Hat 上的 openvpn 软件包将安装适当的启动脚本，可以用该脚本来启动、停止以及重启 OpenVPN 服务器。在 Red Hat 上，用 service 命令启动 OpenVPN 服务器。

```
$ sudo service openvpn start
```

用 chkconfig 命令确保它开机自启动。

```
$ sudo chkconfig --add openvpn
```


在 Ubuntu 上，用 `invoke-rc.d` 命令启动和停止 OpenVPN 服务器，并且用 `update-rc.d` 命令确保它开机自启动。

```
$ sudo invoke-rc.d openvpn start
$ sudo update-rc.d openvpn defaults
```

当 OpenVPN 服务器启动时，它将搜索 `/etc/openvpn` 目录，查找所有后缀为 `.conf` 的文件，并从这些文件中加载所有找到的 VPN。比如，在总公司的服务器上，可能有 `headoffice.conf` 和 `mobileuser.conf` 这两个 OpenVPN 配置文件。启动 OpenVPN 时，这两个文件将被自动加载，并且 OpenVPN 服务器会尝试启动这两个文件中指定的 VPN。

14.2.3 配置 OpenVPN

正如之前所提到过的，需要在任意连接的两端配置 OpenVPN。我们通过在总公司的防御主机上建立 OpenVPN 服务器开始配置过程，然后配置到分公司的连接。这种两公司网络之间的连接称作静态 VPN 或者点对点 VPN。最后，介绍如何为移动用户配置客户端，比如在便携式电脑或者台式机上配置客户端。

1. 推荐的 VPN 配置

接下来快速浏览一下本章推荐的 VPN 通道配置的网络示意图（见 14-2）。下面将在总公司的网关和每个分公司的网关之间建立通道。

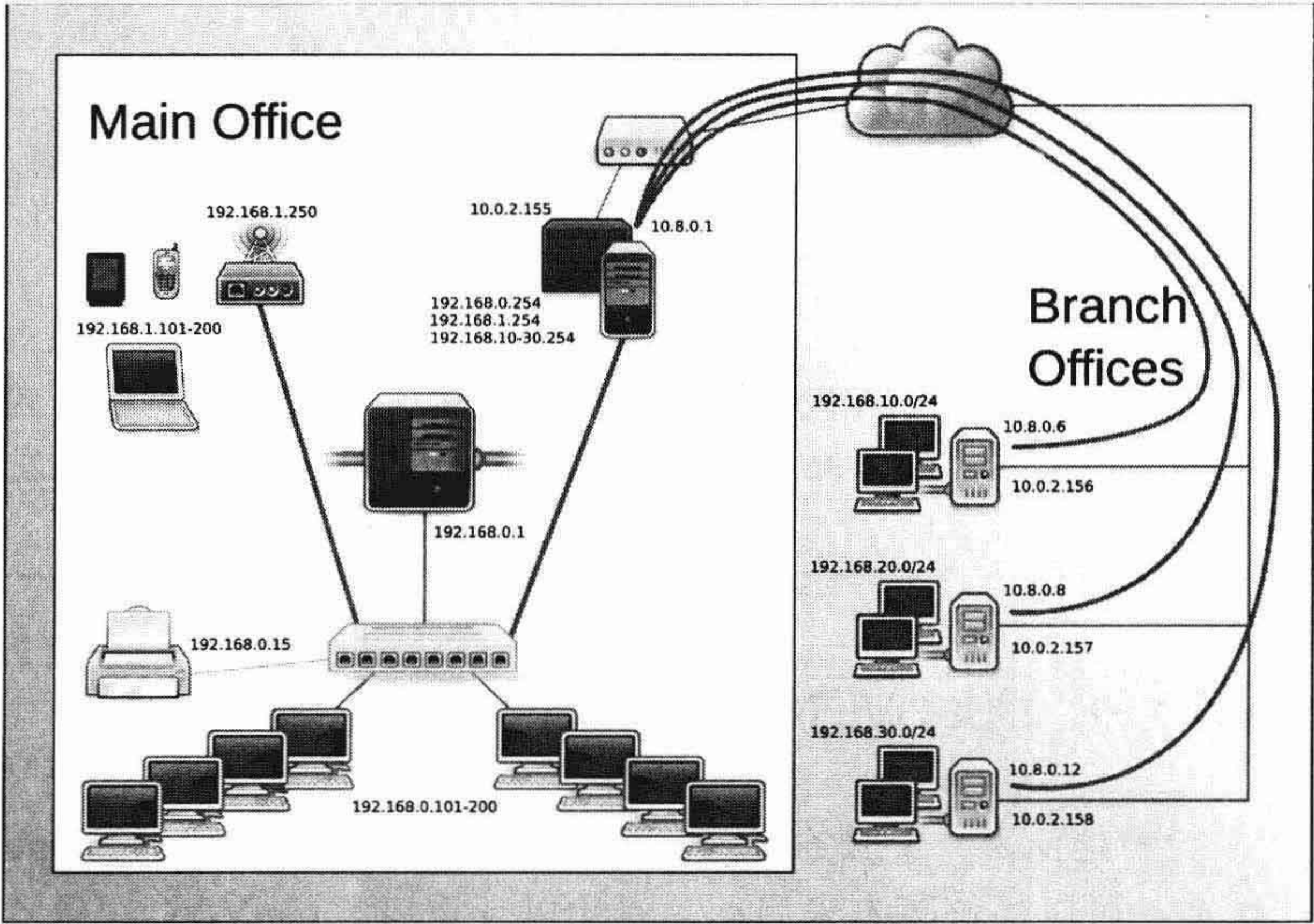


图 14-2 总公司和分公司网络之间的点对点 VPN 配置

我们在总公司的防御主机 `gateway.example.com` 上安装 OpenVPN 服务器，现在开始配置 OpenVPN。首先为 OpenVPN 设置一些基本配置，即在 `/etc/openvpn` 目录下创建一个名为 `gateway.conf` 的文件，该文件如列表 14-1 所示。

列表 14-1 gateway.conf 配置文件

```
# Network configuration
dev tun
port 1194
proto udp
server 10.8.0.0 255.255.255.0
keepalive 10 120

# Logging configuration
log-append /var/log/openvpn.log
status /var/log/openvpn-status.log
verb 4
mute 20

# Security configuration
user nobody
group nobody
persist-key
persist-tun

# Compression
comp-lzo
```

■注：接下来的演示中会用一些附加选项补充 gateway.conf 文件。

列表 14-1 中指定了一些选项。第一个选项是 dev tun，它告诉 OpenVPN 创建一个通道设备和一个虚拟设备 tun0，其中虚拟设备 tun0 用来为 VPN 连接服务。通过指定 dev tun，我们还创建了一个路由模式的 VPN。OpenVPN 可以创建两种 VPN：路由模式的 VPN 和网桥模式的 VPN。简单地说，网桥模式的 VPN 在以太网级别把网络连接在一起，而路由模式的 VPN 依靠 TCP/IP 网络协议把网络连接在一起。本章将使用路由模式的 VPN，因为其可扩展性通常要好一点，而且更适合我们的需求。

■注：如果有兴趣了解更多关于路由模式的 VPN 和网桥模式的 VPN 之间的区别，可以在 <http://openvpn.net/index.php/documentation/howto.html#vpntype> 上找到更多的信息。

接着，列表 14-1 指定了 port 1194 和 proto udp 两个选项。这些选项告诉 OpenVPN 在创建 VPN 连接的时候监听 UDP 端口 1194 上的通信请求。还需要配置防火墙，使其允许该端口上的通信，并接受外部连接。

■注：如果需要，可以修改这些选项，改变端口和协议（TCP）。

通过使用 Netfilter 防火墙和 iptables 命令，可以确保 VPN 的通信穿越相应的端口，如列表 14-2 所示。

列表 14-2 OpenVPN 防火墙规则

```
-A INPUT -i tun0 -p udp -m udp -dport 1194 -d 0/0 -j ACCEPT
```


这里添加一条规则允许 tun0 接口 UDP 端口 1194 上的通信进入系统。
或者使用 TCP，类似以下的规则。

```
-A INPUT -i tun0 -p tcp -m tcp -dport 1194 -d 0/0 -j ACCEPT
```

■注：VPN 配置失败最常见的原因是防火墙的问题。应该经常用 tcpdump 或 nmap 命令检查防火墙是否有允许访问该端口的规则。第 6 章讨论了防火墙的创建、规则和故障排除。

接着，在列表 14-1 中，server 选项列出服务器的 IP 地址和对 VPN 客户端有效的 IP 地址列表。该选项指定了默认网络地址域 10.8.0.0/24。默认情况下，OpenVPN 服务器将占用 IP 地址 10.8.0.1，并把剩下地址分配给进入的 VPN 连接。

下面一个选项是 keepalive，该选项用于保持连接。同时，keepalive 选项被指定了两个值，即 10 和 120。10 指出 OpenVPN 将每 10 秒查验一次连接，查看它是否有效。120 表示 OpenVPN 将最多为一次响应等待 120 秒。如果在 120 秒内没有接收到响应，OpenVPN 将认定该连接已经中断，并通知用户。

接着添加一些日志功能。第一个是 log-append，它告诉 OpenVPN 把日志记录到 /var/log/openvpn.log 文件。第二个选项是 status，它指定一个状态日志文件。OpenVPN 会把当前连接的状态信息输出到该日志文件。在本例中，该文件是 /var/log/openvpn-status.log。第三个选项 verb 告诉 OpenVPN 记录多少日志信息。它包括 0 到 15 的整数，这里 0 代表不记录日志，而 15 代表记录最多。本例中的值为 4，代表 OpenVPN 将生成充分的满足大部分要求的日志信息。

配置选项的下一个区块为 OpenVPN 服务器提供一些附加的安全措施。前面两个选项 user 和 group 允许为 OpenVPN 指定一个用户和组。这样做会限制 OpenVPN 进程所拥有的一些权限（比如 root 用户权限），并且当有某些人要侵入 OpenVPN 进程时，这将确保他没有足够的权限扩大危害。在 Red Hat 主机上，使用 nobody 用户和 nobody 用户组。在 Ubuntu 主机上，推荐使用 nobody 用户和 nogroup 用户组。

```
user nobody  
group nogroup
```

下面两个选项 persist-tun 和 persist-key 与前面权限的限制有关，它们允许 OpenVPN 保留足够的权限使用网络接口和 SSL 证书。

文件中最后的选项 comp-lzo 告诉 OpenVPN 为 VPN 通道使用压缩技术。这将提高通道通信的性能。

现在已经为 VPN 做好了基本的配置，但是还要做身份认证才能保证 OpenVPN 正常、安全地服务。

2. 配置 OpenVPN 身份认证

身份认证能确保只有授权的主机才能发起 VPN 和连接 VPN。OpenVPN 支持多种身份认证方法，包括预设共享密钥、令牌和 TLS/SSL 认证这样的双重身份认证。

最基本的身份认证方法是预设共享密钥。预设共享密钥是静态密钥，它产生于服务端主

机，然后被分配给想要连接的主机。可以用 `openvpn` 命令的 `--genkey` 选项产生一个静态密钥，比如：

```
$ sudo openvpn --genkey --secret /etc/openvpn/secret.key
```

这将在 `/etc/openvpn` 目录下创建一个包含密钥的文件 `secret.key`。这里的文件名和目录已被 `--secret` 选项指定。读者也可以用 `secret` 选项在你的 OpenVPN 配置中指定该文件的位置。

```
secret /etc/openvpn/secret.key
```

接着拷贝该文件——最好采用安全的拷贝方法，比如 `scp`、`SMIME` 或 `GPG/PGP` 加密的 `e-mail`——到其他主机，并把该文件添加到客户端的 OpenVPN 配置中。

但是，我们并不打算使用预设共享密钥的身份认证方法，因为它有一些限制。其最大限制是它只能用于一个服务器和客户端的连接（比如，每个 VPN 通道只能用来连接一个主机）。所以这不是一个理想的模型。举个例子，如果需要连接多个主机或者公司网络，这种方法将不能有效支持多个移动用户连接到总公司的网络。

相反，我们用证书来确保总公司和分公司之间的 VPN 安全。要想使用证书，需要用认证授权中心（Certificate Authority, CA）创建和分配证书签名。

■注：第 10 章重点讨论 CA 过程（如果需要，现在转回第 10 章）。

可以购买证书，或者使用已经创建好的 CA 创建自己的证书，并给它签名。接下来介绍如何使用第 10 章创建的 CA 创建自己的证书，并给它签名。

首先需要为 VPN 服务器创建一个证书服务器。创建新证书过程的第一步是创建一个证书签名请求（certificate signing request, CSR）和一个密钥。现在就开始创建证书的第一步。

```
$ openssl req -new -newkey rsa:2048 -nodes -keyout gateway.example.com.key ➡
-out gateway.example.com.req
```

以上命令生成一个 2048 位的 RSA 密钥，并且创建一个 CSR。系统提示我们输入一些必需的字段（国家、城市等），在这里要采用和认证授权相同的值。还应该把 “Common Name” 字段设置为服务器的全局域名，在本章的例子中是 `gateway.example.com`。

然后需要使用 CA 给证书签名，如列表 14-3 所示。

列表 14-3 给服务器证书签名

```
$ cd /etc/CA
$ sudo openssl ca -out gateway.example.com.cert -config ./openssl.cnf ➡
-infiles gateway.example.com.req
Using configuration from ./openssl.cnf
Enter pass phrase for ./private/cakey.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
    Serial Number: 1 (0x1)
    Validity
        Not Before: Jan 12 09:57:43 2009 GMT
```



```
Not After : Jan 12 09:57:43 2010 GMT
Subject:
  countryName      = AU
  stateOrProvinceName = Victoria
  organizationName  = Example Pty Ltd
  commonName       = gateway.example.com
  emailAddress      = postmaster@example.com
X509v3 extensions:
  X509v3 Basic Constraints:
    CA:FALSE
  Netscape Comment:
    OpenSSL Generated Certificate
  X509v3 Subject Key Identifier:
    8F:A6:48:91:D5:3F:70:7B:E3:E2:AB:E5:F5:41:8A:F4:20:DA:31:6E
  X509v3 Authority Key Identifier:
    keyid:F7:0A:17:47:FA:0D:7B:C4:FA:63:0C:C9:FC:5B:49:1C:D5:C3:FF:1D
```

```
Certificate is to be certified until Jan 12 09:57:43 2010 GMT (365 days)
Sign the certificate? [y/n]:y
```

```
1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

首先，切换到/etc/CA 目录，然后运行“openssl ca”命令给 CSR 签名，这将输出一个 CA 签名的证书。现在有了两个文件，gateway.example.com.key（私钥文件）和 gateway.example.com.cert（证书文件），把这两个文件移动到/etc/openvpn 目录。

```
$ sudo mv gateway.example.com.* /etc/openvpn
```

■注：除私钥文件和证书文件之外，还有一个 CSR 请求文件。保留它是有用的，因为当证书失效时，它将派上用场。可以再次用该请求文件创建新的证书，正如第 10 章提到过的。

赋予密钥和证书正确的所有权和权限限制来保护它们。

```
$ sudo chown root:root /etc/openvpn/gateway.example.com.*
$ sudo chmod 0600 /etc/openvpn/gateway.example.com.key
```

还需要创建一些 Diffie-Hellman 参数。Diffie-Hellman 参数是可以加强 VPN 会话安全的加密参数（读者可以在 <http://www.rsa.com/rsalabs/node.asp?id=2248> 上详细阅读关于这些参数的内容）。可以用 openssl 命令创建 Diffie-Hellman 参数。

```
$ sudo openssl dhparam -out /etc/openvpn/dh2048.pem 2048
Generating DH parameters, 2048 bit long safe prime, generator 2
This is going to take a long time
.....+.....
```

这里，在/etc/openvpn 目录下创建一个文件 dh2048.pem。它是一个 2048 位的 DH 参数文件，命令行中的参数 2048 用来指定其文件大小。还需要创建一个和证书文件相同大小的文件，其大小是 2048 位，正如之前证书请求所产生的文件大小。

现在，通知 OpenVPN 新的 CA 证书以及 DH 参数文件的位置（见列表 14-4）。

列表 14-4 gateway.conf 配置文件

```
# Network configuration
dev tun
port 1194
proto udp
server 10.8.0.0 255.255.255.0
keepalive 10 120

# Certificate configuration
ca /etc/CA/cacert.pem
dh /etc/openvpn/dh2048.pem
cert /etc/openvpn/gateway.example.com.cert
key /etc/openvpn/gateway.example.com.key

# Logging configuration
log-append /var/log/openvpn.log
status /var/log/openvpn-status.log
verb 4
mute 20

# Security configuration
user nobody
group nobody
persist-key
persist-tun

# Compression
comp-lzo
```

读者会注意到我们添加了 4 个选项。第一个是 `ca` 选项，它用来指定 CA 证书的位置，在例子中它是 `/etc/CA/cacert.pem`。下一个选项 `dh` 指定了 Diffie-Hellman 参数文件的位置，在本例子中是 `/etc/openvpn/dh2048.pem`。最后，用 `cert` 和 `key` 选项分别指定证书文件和密钥文件的位置，例子中它们分别是 `/etc/openvpn/gateway.example.com.cert` 和 `/etc/openvpn/gateway.example.com.key`。

在 Ubuntu 主机上，可以按照以下的方法重启 OpenVPN 服务器。

```
$ sudo invoke-rc.d openvpn restart
* Stopping virtual private network daemon(s) ...
  * Stopping VPN 'gateway' [ OK ]
* Starting virtual private network daemon(s) ...
  * Autostarting VPN 'gateway' [ OK ]
```

可以看到已经重启了 OpenVPN 服务器，并且在它重启的时候，它还自动启动了一个叫做“gateway”的 VPN。该 VPN 以 `gateway.conf` 配置文件的文件名命名的。

我们还可以在 `/var/log/openvpn.log` 文件中看到一些日志记录项，如列表 14-5 所示。

列表 14-5 /var/log/openvpn.log 日志文件

```
Fri Jan 30 00:36:32 2009 OpenVPN 2.1_rc11 i486-pc-linux-gnu [SSL] [LZO2]
[EPOLL] [PKCS11] built on Oct 15 2008
Fri Jan 30 00:36:32 2009 /usr/bin/openssl-vulnkey -q -b 2048 -m <modulus omitted>
```



```
Fri Jan 30 00:36:32 2009 TUN/TAP device tun0 opened
Fri Jan 30 00:36:32 2009 /sbin/ifconfig tun0 10.8.0.1 pointopoint 10.8.0.2 mtu 1500
Fri Jan 30 00:36:32 2009 UDPv4 link local (bound): [undef]:1194
Fri Jan 30 00:36:32 2009 UDPv4 link remote: [undef]
Fri Jan 30 00:36:32 2009 Initialization Sequence Completed
```

列表 14-5 显示 OpenVPN 服务器已经启动，网络接口 tun0 已创建，IP 地址 10.8.0.1 已添加并绑定到 UDP 端口 1194。

■注：要确保添加正确的防火墙规则允许 VPN 连接。在本例中，需要接受在 UDP 端口 1194 上的外来连接。

虽然服务器已经运行，但是还需要配置客户端，并为客户端安装 OpenVPN 软件和创建用于连接的证书。

■注：继续讲解本章会涉及两个主机。当需要在特定的主机上执行命令的时候，后面的部分会在命令行前面加个上主机名，把它作为前缀，如 gateway\$ 或者 branch1\$。

3. 在分公司服务器上配置 OpenVPN

首先需要在分公司服务器 branch1.example.com 上安装 OpenVPN 软件包，然后根据客户端的 Linux 版本，用相应的命令设置 OpenVPN 开机自启动。比如，在 Red Hat 上：

```
branch1$ sudo chkconfig --add openvpn
```

在 Ubuntu 上：

```
branch1$ sudo update-rc.d openvpn defaults
```

■注：上面的命令都使用了命令行前缀 branch1\$，读者应该知道我们正在 branch1.example.com 主机上运行上面的命令。

下一步为每个分公司创建证书和密钥。即在 gateway.example.com 主机上创建证书和密钥，然后用该主机上的 CA 给证书和密钥签名。首先配置一个分公司服务器 branch1.example.com。

```
gateway$ openssl req -new -newkey rsa:2048 -nodes -keyout branch1.example.com.key -out branch1.example.com.req
```

这里生成一个 2048 位的 RSA 密钥并且创建了一个 CSR。系统会提示输入一些必需的字段，如国家、城市等，并且必须使用和认证授权相同的值。还应该把 “Common Name” 字段设置为服务器的全局域名，本例子中它是 branch1.example.com。

接着，按照之前的做法用 CA 给证书签名。

```
gateway$ cd /etc/CA
gateway$ sudo openssl ca -out branch1.example.com.cert -config ./openssl.cnf -infiles branch1.example.com.req
```

现在，需要给分公司服务器发送证书和密钥以及 CA 证书。有多种方法可以用来发送证书等文件，但必须安全地发送它们——比如，不要用 e-mail 发送它们，因为它们会被轻易拦截。

截。可以把这些文件存放在 USB key 或者 DVD 上，然后本地安装它们。如果能够连接主机，就可以使用 `scp`（安全拷贝）命令或者 `sftp`（安全的 FTP）命令发送这些文件（`scp`、`sftp` 这些命令使用 SSH 安全地连接另一个主机并传输文件）。

下面用 `sftp` 命令连接分公司服务器并传输所需文件。

```
gateway$ sftp jsmith@branch1.example.com
Connecting to branch1.example.com...
jsmith@branch1.example.com's password:
sftp> put branch1.example.com.cert
Uploading /home/jsmith/branch1.example.com.cert to
/home/jsmith/branch1.example.com.cert
/home/jsmith/branch1.example.com.cert 100% 5881 12.0KB/s 00:03
```

上面的命令首先连接到 `branch1.example.com` 主机，然后通过 `put` 命令把 `branch1.example.com.cert` 证书文件从 `/home/jsmith` 目录传输到 `branch1` 主机上对应的目录 `/home/jsmith`。在远程主机上，可以用 `cd` 命令切换当前的目录。

```
sftp> cd /tmp
```

这里需要具有写目录的权限。在本地主机 `gateway.example.com` 上，如果目标文件存放在别处，可以用 `lcd` 命令把当前目录切换到目标文件所在的目录。

我们再用 `put` 命令把 `branch1.example.com.key` 和 CA 证书文件 `cacert.pem` 发送到 `branch1` 主机。现在，把这些文件移动到 `/etc/openvpn` 目录，并给它们赋予安全的所有权和访问权限。

```
branch1$ sudo mv branch1.example.com.cert /etc/openvpn
branch1$ sudo mv branch1.example.com.key /etc/openvpn
branch1$ sudo mv cacert.pem /etc/openvpn
```

下面的代码赋予证书正确的所有权和权限限制，以便保护它们。

```
branch1$ sudo chown root:root /etc/openvpn/branch1.example.com.*
branch1$ sudo chown root:root /etc/openvpn/ca.cert
branch1$ sudo chmod 0600 /etc/openvpn/gateway.example.com.key
```

下一步，为客户端创建配置文件。在 `/etc/openvpn` 目录下创建一个名叫 `branch1.conf` 的文件，如列表 14-6 所示。

列表 14-6 branch1.conf 配置文件

```
# Network configuration
dev tun
client
remote gateway.example.com 1194
keepalive 10 120

# Certificate configuration
ca /etc/openvpn/cacert.pem
cert /etc/openvpn/branch1.example.com.cert
key /etc/openvpn/branch1.example.com.key

# Logging configuration
log-append /var/log/openvpn.log
status /var/log/openvpn-status.log
verb 4
```



```
mute 20

# Security configuration
user nobody
group nobody
persist-key
persist-tun

# Compression
comp-lzo
```

■注：在 Ubuntu 主机上，group 设置项应该设置为 nogroup，而不是 nobody。

列表 14-6 的文件和 gateway.conf 配置文件非常相似，但是因为该主机的角色是一个客户端，所以指定了一些不同的选项。再次为路由模式的 VPN 指定 dev tun。同时，指定 client 选项和 remote 选项。client 选项指明该主机是一个客户端，remote 选项告诉 OpenVPN 从哪里连接 VPN 通道。这里，这两个选项被指定为 gateway.example.com 和 port 1194。

■注：OpenVPN 必须能辨别主机（比如，它必须可以找到主机的一个 IP 地址）。如果没有 DNS 服务器（应该有一个），则可以直接为该选项指定一个 IP 地址。

4. 启动分公司服务器上的 OpenVPN

既然 VPN 已经配置完成，那么现在就可以启动客户端 OpenVPN 程序。在 Red Hat 上用以下命令：

```
branch1$ sudo service openvpn start
```

而在 Ubuntu 上用以下命令：

```
branch1$ sudo invoke-rc.d openvpn start
```

5. 测试 OpenVPN 通道

可以用多种方法测试 VPN 连接是否可用，下面将带领读者依次尝试这些方法。首先，查看主机 branch1 的日志文件 /var/log/openvpn.log 中的一些记录项，可以看到一些和列表 14-5 相似的记录项，而且还会看到客户端连接服务器的会话过程。

```
Fri Jan 30 18:51:35 2009 us=406059 [gateway.example.com] Peer Connection Initiated with 10.0.2.155:1194
```

查看主机 gateway 的 /var/log/openvpn.log 文件中的一些记录项。这些记录项显示了连接过程。

```
gateway$ less /var/log/openvpn.log
Fri Jan 30 18:50:19 2009 us=518613 10.0.2.156:1194 [branch1.example.com] Peer Connection Initiated with 10.0.2.155:1194
Fri Jan 30 18:50:19 2009 us=518706 branch1.example.com/10.0.2.156:1194 MULTI: Learn: 10.8.0.6 -> branch1.example.com/10.0.2.156:1194
```


另外，可以看到 gateway 主机和 branch1 主机创建了新的网络接口，其接口名字以 tun 开头。在 gateway 主机上，可以看到一个叫做 tun0 的新的网络接口，其 IP 地址是 10.8.0.1（正如之前所提到过的）。

```
gateway$ ifconfig -a
tun0      Link encap:UNSPEC HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:10.8.0.1 P-t-P:10.8.0.2 Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1500 Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
```

在 branch1 主机上也创建了一个叫做 tun0 的网络接口，该网络接口的 IP 地址是 10.8.0.6，来自服务器提供的 IP 地址列表。

```
branch1$ ifconfig -a
tun0      Link encap:UNSPEC HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:10.8.0.6 P-t-P:10.8.0.5 Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1500 Metric:1
          RX packets:293 errors:0 dropped:0 overruns:0 frame:0
          TX packets:300 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:24612 (24.0 KiB) TX bytes:25200 (24.6 KiB)
```

还可以看到 branch1 主机的路由表（用第 6 章介绍的 ip 命令）中有一条到 10.8.0.0/24 网络的线路。

```
branch1$ ip route show
10.8.0.2 dev tun0 proto kernel scope link src 10.8.0.1
10.8.0.0/24 via 10.8.0.2 dev tun0
192.0.2.0/24 dev eth1 proto kernel scope link src 192.0.2.155 metric 1
192.168.122.0/24 dev vnet0 proto kernel scope link src 192.168.122.1
169.254.0.0/16 dev eth0 proto kernel scope link src 169.254.6.35
default via 192.0.2.2 dev eth1 proto static
default dev eth0 scope link metric 1000
```

从上面的路由信息可以看出，这里有一条通过 tun0 接口连接到 10.8.0.1 主机的线路。

注意到两个主机都创建了一个叫做/var/log/openvpn-status.log 的文件，该文件包含当前连接的一个列表，并且每 60 秒更新一次。接下来查看主机 gateway 的/var/log/openvpn-status.log 文件。

```
gateway$ less /var/log/openvpn-status.log
OpenVPN CLIENT LIST
Updated,Fri Jan 30 20:37:15 2009
Common Name,Real Address,Bytes Received,Bytes Sent,Connected Since
branch1.example.com,10.0.2.156:1194,118186,117874,Fri Jan 30 18:50:19 2009
ROUTING TABLE
Virtual Address,Common Name,Real Address,Last Ref
10.8.0.6,branch1.example.com,10.0.2.156:1194,Fri Jan 30 20:37:14 2009
GLOBAL STATS
Max bcast/mcast queue length,0
END
```


可以看到这里列出了一条来自主机 branch1 (IP 地址是 10.0.2.156) 的连接。

最后, 还可以用一些网络工具测试实际的连接。首先在 gateway 主机上用 ping 命令测试和 branch1 主机的连通性。下面 ping 一下 branch1 主机上的 VPN 通道端点的地址 (该地址是 10.8.0.6)。

```
gateway$ ping 10.8.0.6
PING 10.8.0.6 (10.8.0.6) 56(84) bytes of data.
64 bytes from 10.8.0.6: icmp_seq=1 ttl=64 time=0.800 ms
64 bytes from 10.8.0.6: icmp_seq=2 ttl=64 time=0.756 ms
64 bytes from 10.8.0.6: icmp_seq=3 ttl=64 time=2.26 ms
64 bytes from 10.8.0.6: icmp_seq=4 ttl=64 time=1.38 ms
64 bytes from 10.8.0.6: icmp_seq=5 ttl=64 time=1.17 ms
```

上述内容显示 gateway 主机可以通过 ICMP 协议到达 branch1 主机, 而且 branch1 主机 (IP 地址是 10.8.0.6) 也应答了 ICMP 请求。

我们还可以做同样的事情, 在 branch1 主机上尝试 ping 一下 gateway 主机端点的 IP 地址 10.8.0.1。

```
branch1$ ping 10.8.0.1
PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data.
64 bytes from 10.8.0.1: icmp_seq=1 ttl=64 time=10.02 ms
64 bytes from 10.8.0.1: icmp_seq=2 ttl=64 time=20.21 ms
64 bytes from 10.8.0.1: icmp_seq=3 ttl=64 time=10.19 ms
64 bytes from 10.8.0.1: icmp_seq=4 ttl=64 time=20.51 ms
64 bytes from 10.8.0.1: icmp_seq=5 ttl=64 time=10.17 ms
```

如果两个端点都应答了 ICMP 请求, 则说明了 VPN 通道已经可用, 可以用来转发分公司网络和总公司网络之间的通信。

如果需要, 还可以为其他分公司网络重复该配置过程。比如, 在本例中, 还可以添加网络 branch2.example.com 和总公司网络之间的 VPN 通道, 以及网络 branch3.example.com 和总公司网络之间的 VPN 通道。

14.2.4 用 OpenVPN 发布总公司资源

用目前的配置就可以在总公司和分公司的网络之间使用 VPN 通道转发通信。下面看一下总公司的网络和分公司的网络是如何交互的。

现在有两条从分公司到总公司的路径。第一条路径是在第 6 章看到的经由 192.0.2.0/24 网络的路径, 它是在公司和 Internet 之间建立的 DSL 或者 ASDL (或者其他相似的) Internet 连接。每个独立的子公司通常拥有各自的 Internet 连接。我们已把 192.0.2.0/24 网络用作它们的 IP 地址, 很可能每个子公司都拥有从 ISP (Internet 服务提供者) 获得的独立的 IP 地址。

网络 192.0.2.0/24 并不安全, 因为它是运行在 Internet 之上的。除非确保特定的应用程序或者协议是安全的 (比如, 第 10 章介绍过使用 SSL/TLS 方法来保护 SMTP 和 IMAP 服务), 否则攻击者就可以读懂我们在 Internet 网络上的数据。由于这个潜在的安全问题, 需要在这些主机和 10.8.0.0/24 网络 (做 VPN 通道之用) 之间的 Internet 连接上创建第二条传输路径。总公司有 OpenVPN 服务器, 其 IP 地址是 10.8.0.1。每个分公司配有该网络域中的一个 IP 地

址——比如，branch1.example.com 配有 IP 地址 10.8.0.6。

然而，目前只能通过 VPN 通道连到 IP 地址 10.8.0.1。这对我们没多大用处，因为还不能访问该内部网络中的资源（比如 headoffice.example.com 主机上的共享资源）。我们可以在分公司的主机上试着 ping 总公司的主机来测试一下。

```
branch1$ ping 192.168.0.1
```

显然得不到 ping 请求的应答，因为还没有到达该网络地址的路径。为了访问这些主机上的资源，要依次确保两件事情：路由和防火墙规则。接下来的几节会介绍这些内容。

1. 路由

首先，需要配置分公司网络，使其能连接到总公司的内部网络。为此，设置分公司网络，当它要连接到 192.168.0.0/24 网络时，需要通过 VPN 通道连接 gateway 主机。如列表 14-7 所示，在 gateway 主机上的 gateway.conf 配置文件中添加一行，增加一条到分公司的路由通道。

列表 14-7 在 gateway.conf 文件中添加推式路由

```
push "route 192.168.0.0 255.255.255.0"
```

这一行给连接到 OpenVPN 服务器的所有客户端添加一条路由项。为了使新的路由项生效，需要重启主机 gateway 和分公司主机的 OpenVPN 服务。

如果看一下 branch1 主机上的路由表，就会在路由表中看到一条到 192.168.0.0/24 网络的新路由项（加粗显示的那条）。

```
branch1$ route
Kernel IP routing table
Destination  Gateway      Genmask      Flags        Metric     Ref    Use  Iface
10.8.0.5      *            255.255.255.255  UH          0          0     0   tun0
10.8.0.1      10.8.0.5    255.255.255.255  UGH         0          0     0   tun0
192.168.0.0  10.8.0.5    255.255.255.0   UG          0          0     0   tun0
192.0.2.0     *            255.255.255.0   U           0          0     0   eth1
169.254.0.0   *            255.255.0.0     U           0          0     0   eth1
default       192.0.2.2   0.0.0.0         UG          0          0     0   eth1
```

现在可以在主机 branch1 上 ping 192.168.0.0/24 网络了。

```
branch1$ ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=64 time=1.18 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=64 time=1.31 ms
64 bytes from 192.168.0.1: icmp_seq=3 ttl=64 time=2.33 ms
64 bytes from 192.168.0.1: icmp_seq=4 ttl=64 time=1.25 ms
64 bytes from 192.168.0.1: icmp_seq=5 ttl=64 time=0.923 ms
```

可以看到 branch1 主机获得了来自 192.168.0.1 主机的应答消息。这意味着只要 gateway 主机上的防火墙允许，我们就可以访问 192.168.0.0/24 网络上的资源了。

如果 branch1 主机是 192.168.10.0/24 网络（branch1 所在的局域网）的默认路由，那么 192.162.10.0/24 网络中的所有用户都能够访问总公司 192.168.0.0/24 网络中的资源。

■注：本章不打算讲解如何让总公司网络路由到分公司的内部网络，但是这也是可行的。请读者访问 “<http://www.secure-computing.net/wiki/index.php/OpenVPN/Routing>”，找到该配置的例子。

2. 防火墙

我们还需要设置 gateway 主机和分支网络主机上的防火墙，确保防火墙允许相关的网络通信。在 gateway 主机上，需要把 IP 通信从 gateway 主机转发到内部网，以及把来自内部网络的 IP 通信转发出去。这很像第 6 章中对防御主机 gateway.example.com 转发服务所做的配置。

首先，对于通过 gateway 主机的 VPN 通道接口 tun0 转发出去的通信，直接允许它穿过自己的防火墙链。用下面方式指定该链为 Firewall-tun0-FORWARD。

```
-A FORWARD -i tun0 -o eth0 -j Firewall-tun0-FORWARD
```

接着，为该链创建一个规则集，允许 gateway 主机转发特殊的通信，如列表 14-8 所示。

列表 14-8 为 OpenVPN 路由所建的一些 IP 表规则样例

```
-A Firewall-tun0-FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
-A Firewall-tun0-FORWARD -p tcp -m state --state NEW -m tcp --dport 25 -j ACCEPT
-A Firewall-tun0-FORWARD -p tcp -m state --state NEW -m tcp --dport 53 -j ACCEPT
-A Firewall-tun0-FORWARD -p udp -m state --state NEW -m udp --dport 53 -j ACCEPT
-A Firewall-tun0-FORWARD -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
-A Firewall-tun0-FORWARD -p udp -m state --state NEW -m udp --sport 123 --dport 123 -j ACCEPT
-A Firewall-tun0-FORWARD -p tcp -m state --state NEW -m tcp --dport 443 -j ACCEPT
-A Firewall-tun0-FORWARD -p tcp -m state --state NEW -m tcp --dport 993 -j ACCEPT
-A Firewall-tun0-FORWARD -j REJECT --reject-with icmp-port-unreachable
```

这里创建了多种简单的规则，用来允许 VPN 通道上的通信，并把它转发到内部网络。现在，VPN 通道已经能为 SMTP、HTTP/HTTPS、IMAP 以及其他协议的数据做转发了。

■注：可以用第 6 章提供的命令为 gateway 主机添加这些防火墙规则。

14.2.5 为移动用户创建 VPN 连接

OpenVPN 具有很高的可扩展性，不仅支持从分公司到总公司之间点对点的连接，还允许移动用户连接总公司网络并访问共享文件、打印机和应用程序等资源。为了实现移动用户访问总公司网络，需要在 gateway 主机上建立另一个 VPN 通道，并在移动用户客户端上安装 OpenVPN。本章之前已提到过，OpenVPN 已经可以在 Linux、Microsoft Windows、Mac OS X 和其他一些平台上运行。

对于移动用户而言，建立 VPN 连接会有些不同。我们不会使用证书（即使可以使用它）来对客户端做身份认证，因为生成大量证书的费用可能会非常高。

■注：有个能简化 OpenVPN 证书管理的工具，它是 easy-rsa。Red Hat 和 Ubuntu 上都有该工具。在 Red Hat 主机上，其位置是 /usr/share/doc/openvpn-version/easy-rsa；在 Ubuntu 主机上，其位置是 /usr/share/doc/openvpn/examples/easy-rsa。

相反，下面将介绍如何使用 PAM（第 4 章介绍过）对用户做身份认证。由于 PAM 能够签入多种认证机制，可以用它包含以下一些认证机制。

- 本地 Linux 用户。
- RSA 令牌和智能卡这样的双重认证。
- Kerberos。
- RADIUS。
- IMAP（不是 IMAP 服务器）。
- LDAP。

下面会介绍如何配置基本的本地用户认证。也就是客户端用户在 gateway 主机上拥有一个 Linux 用户账户，并可以通过身份认证，就如同通过控制台或 SSH 登录到 gateway 主机一样。可以使用 PAM 方便地扩展到其他形式的身份认证。

1. 为移动用户配置 VPN

我们需要一个新的.conf 文件开始配置过程。在/etc/openvpn 目录下创建一个叫做 mobile.conf 的文件，如列表 14-9 所示。

列表 14-9 移动用户的配置文件 mobile.conf

```
# Network configuration
dev tun
port 1195
proto udp
server 10.9.0.0 255.255.255.0
keepalive 10 120

# Certificate configuration
dh /etc/openvpn/dh2048.pem
ca /etc/CA/ca.cert
cert /etc/openvpn/gateway.example.com.cert
key /etc/openvpn/gateway.example.com.key
plugin /usr/lib/openvpn/openvpn-auth-pam.so passwd
client-cert-not-required
username-as-common-name

# Logging configuration
log-append /var/log/openvpn-mobile.log
status /var/log/openvpn-status-mobile.log
verb 4
mute 20

# Security configuration
user nobody
group nogroup
persist-key
persist-tun

# Compression
comp-lzo
```


■注：当本节涉及在客户端上配置路由和相关功能的时候，将对该配置文件展开讨论。

可以看到，配置文件 `mobile.conf` 和 VPN 通道的配置文件 `gateway.conf` 很相似。重点看一下不同的地方。在 `mobile.conf` 文件中，更改了一些网络配置，即使用不同的端口号 1195，因为 1194 端口已被其他 VPN 通道绑定。另外，为移动用户指定附加的 IP 子网，如 10.9.0.0/24。

我们还需要在防火墙中为该子网添加合适的规则。首先，在 `tun1` 网络接口上为该通道打开 1195 端口（新通道的网络接口编号已经增加了）。

```
-A INPUT -i tun1 -p udp -m udp -dport 1195 -d 0/0 -j ACCEPT
```

■注：因为已经有了一个 `tun0` 网络接口，所以这个网络接口是 `tun1`。如果没有之前那个网络接口，那么该接口可能就是 `tun0`。在 IP 表中，可以用符号 `tun+` 指代所有 `tun` 网络接口。

这里指定相同的证书、密钥以及 DH 参数，不过添加了新的选项 `plugin`。`plugin` 配置选项允许指定外部插件。本章的例子会用到一个叫做 `openvpn-auth-pam.so` 的 PAM 插件，它是和 OpenVPN 软件包捆绑在一起的。

在 Ubuntu 中，该插件位于 `/usr/lib/openvpn/` 目录。为了保证它在 Ubuntu 上正常运行，还需要安装一个附加软件包。

```
$ sudo apt-get install libpam0g-dev
```

在 Red Hat 主机上，`libpam0g-dev` 软件包是不需要的。在 Red Hat 上，所需要的插件位于 `/usr/share/openvpn/plugin/lib` 目录，`plugin` 配置行如下所示。

```
plugin /usr/share/openvpn/plugin/lib/openvpn-auth-pam.so system-auth
```

还需要指定一个 PAM 认证文件，该文件将被 OpenVPN 用于 VPN 通道的身份认证。在 Ubuntu 主机上，指定 PAM 的标准影子口令认证文件 `passwd` 用于身份认证。在 Red Hat 主机上，使用 PAM 的默认认证文件 `system-auth`。

如果要使用其他的身份认证方式，就还要指定一个合适的 PAM 认证文件。举个例子，为了支持双重认证，需要指定一个用于 `pam_rsa.so` 模块（<http://www.rsa.com/node.aspx?id=1177>）的文件。`pam_rsa.so` 模块允许把证书结合到 RSA SecureID 服务器，并且使用 RSA 令牌认证用户。

`smobile.conf` 文件还指定了其他两个选项，即 `client-cert-not-required` 和 `username-ascommon-name`。`client-cert-not-required` 选项禁用移动用户的身份认证；`username-ascommon-name` 把移动用户所提供的用户名作为证书的通用名（CN）。

■注：有些人不喜欢禁用客户端证书，而在实际中他们往往同时使用用户名/密码和证书两种认证。

我们更新了日志文件，为移动连接创建了新的日志文件，同时指定了 `user` 和 `group` 选项（对于本例，Ubuntu 系统上用的是 `nogroup` 用户组）。

接着，重启 OpenVPN 服务器，启用移动 VPN 通道。

2. 配置移动 VPN 客户端

我们还需要配置客户端，好让它连接 gateway 主机。根据不同的客户端，可以使用不同的配置方法。很多客户端工具都可以用于连接 gateway 主机，这些覆盖了从普通的 OpenVPN binary 到高级的 GUI 客户端。本节会提供一些客户端列表，并介绍如何让这些客户端通过 OpenVPN 连接 gateway 主机。

最简单的客户端是 OpenVPN binary。为了用它来连接 gateway 主机，需要有服务器的 CA 证书。如果不打算用证书，就不需要创建该 CA 证书，也不需要把它传递给主机 gateway 和客户端。

如果打算使用 OpenVPN binary，还需要创建一个客户端配置文件。把该文件命名为 mobileclient.conf，并把它保存在客户端的/etc/openvpn 目录下，如列表 14-10 所示。

列表 14-10 mobileclient.conf 配置文件

```
# Network configuration
dev tun
client
remote gateway.example.com 1195
keepalive 10 120

# Certificate configuration
ca /etc/openvpn/ca.cert
auth-user-pass

# Logging configuration
log-append /var/log/openvpn-mobile.log
status /var/log/openvpn-mobile-status.log
verb 4
mute 20

# Security configuration
user nobody
group nogroup
persist-key
persist-tun

# Compression
comp-lzo
```

可以看到这些选项大多在前面的配置文件中出现过，这里只添加一个选项，并对原来的选项做很小的改动。上面的配置把将要连接的远程端口改为 1195，同时添加了 auth-user-pass 选项。auth-user-pass 选项告诉客户端使用用户名和密码的认证方式，而不是证书认证。

■注：正如之前提到过的，如果想同时使用用户名/密码和证书，则需要保留 auth-user-pass 选项，并且添加 cert 和 key 选项。

现在，启动客户端 OpenVPN，它将连接到 gateway 主机，并提示输入合适的认证信息。

```
$ sudo /etc/init.d/openvpn restart
Shutting down openvpn: [ OK ]
```



```
Starting openvpn:
Enter Auth Username:jsmith
Enter Auth Password:*****
```

注意用户名和密码提示符。输入 gateway 主机上的一个用户的用户名 jsmith 和该用户的密码。

接着，客户端将连接上 gateway 主机，并且可以看到一个新的网络接口（例子中它是 tun1）。

```
tun1 Link encap:UNSPEC HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
  inet addr:10.9.0.6 P-t-P:10.9.0.5 Mask:255.255.255.255
  UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1500 Metric:1
  RX packets:0 errors:0 dropped:0 overruns:0 frame:0
  TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
  collisions:0 txqueuelen:100
  RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)
```

该网络接口的 IP 地址 10.9.0.6 是由 gateway 主机分配的（别忘了，移动客户端的 VPN 网络为 10.9.0.0/24）。那么可以 ping 一下 gateway 主机的 IP 地址（例子中，该地址是 10.9.0.1），反之也可以在 gateway 主机上 ping 一下地址 10.9.0.6。

```
$ ping 10.9.0.1
PING 10.9.0.1 (10.9.0.1) 56(84) bytes of data.
64 bytes from 10.9.0.1: icmp_seq=1 ttl=64 time=10.3 ms
64 bytes from 10.9.0.1: icmp_seq=2 ttl=64 time=10.64 ms
64 bytes from 10.9.0.1: icmp_seq=3 ttl=64 time=10.59 ms
64 bytes from 10.9.0.1: icmp_seq=4 ttl=64 time=10.73 ms
64 bytes from 10.9.0.1: icmp_seq=5 ttl=64 time=10.59 ms
```

如果不使用 OpenVPN binary 客户端，则还可以使用其他客户端。补充知识“OpenVPN 客户端”介绍了这些为不同平台提供的 VPN 客户端。

OpenVPN 客户端

有许多支持 OpenVPN 的客户端，它们运行在多个平台上。下面 OpenVPN 客户端的简短列表是按所处平台划分的。

Linux

- KVpnc (KDE): <http://home.gna.org/kvpnc/en/index.html>
- network-manager-openvpn Package (Gnome): <http://packages.ubuntu.com/gutsy/network-manager-openvpn>

Mac OS X

- Tunnelblick: <http://code.google.com/p/tunnelblick/>
- Viscosity: <http://www.viscosityvpn.com/>

Microsoft Windows

- Windows 版 OpenVPN: <http://openvpn.se/>

PocketPC

- PocketPC 版 OpenVPN: <http://ovpnppc.ziggurat29.com/ovpnppc-main.htm>

3. 移动 VPN 路由

现在，可以在总公司和分公司的网络之间创建点对点的 VPN 通道，也能够移动客

户端和 gateway 主机之间完成多种路由配置（创建 VPN 通道）。也就是说，我们不仅把客户端连到处于 gateway 主机之后的主机，而且还告诉客户端如何配置自己，特别是 DHCP 设置。

接下来介绍如何强制让所有通信都通过 VPN 通道。这是一种确保用户通信只限于组织内部的常用方法。举个例子，让所有用户的 Web 通信都经过 Web 代理，这是经常用到的。因此，需要确保它遵循组织认可的使用原则或者其他类似的标准。

首先，我们允许移动用户连到总公司的内部网络 192.168.0.0/24。为此在 gateway 主机的 mobile.conf 配置文件中添加 push 选项，通过它来推送路由。

```
push "route 192.168.0.0 255.255.255.0"
```

接下来还需要更改 gateway 主机的防火墙规则，其方法和我们为分公司和总公司之间的通道添加规则非常相似。下面为移动 VPN 通道添加另一条链。

```
-A FORWARD -i tun1 -o eth0 -j Firewall-tun1-FORWARD
```

接着需要为该链创建一些规则，让 gateway 主机转发特定的通信，如列表 14-11 所示。

列表 14-11 OpenVPN 路由的一些 IP 表规则样例

```
-A Firewall-tun1-FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
-A Firewall-tun1-FORWARD -p tcp -m state --state NEW -m tcp --dport 25 -j ACCEPT
-A Firewall-tun1-FORWARD -p tcp -m state --state NEW -m tcp --dport 53 -j ACCEPT
-A Firewall-tun1-FORWARD -p udp -m state --state NEW -m udp --dport 53 -j ACCEPT
-A Firewall-tun1-FORWARD -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
-A Firewall-tun1-FORWARD -p udp -m state --state NEW -m udp --sport 123 --dport 123 -j ACCEPT
-A Firewall-tun1-FORWARD -p tcp -m state --state NEW -m tcp --dport 443 -j ACCEPT
-A Firewall-tun1-FORWARD -p tcp -m state --state NEW -m tcp --dport 993 -j ACCEPT
-A Firewall-tun1-FORWARD -j REJECT --reject-with icmp-port-unreachable
```

这里创建了多种简单的规则，允许 VPN 通道上的通信，并把它转发到内部网络。现在，VPN 通道已经能为 SMTP、HTTP/HTTPS、IMAP 和其他协议的数据做转发。

■注：可以用第 6 章提供的命令向 gateway 主机添加这些规则。

我们还可以给客户端传递多种选项，比如，帮助客户端设置 DNS 和 WINS 服务器这样的 DHCP 选项。

■注：第 9 章建立 DHCP。

不同种类的客户端（比如 Linux 和 Microsoft Windows 客户端）需要不同的方法来推送所需的选项。当给 Microsoft Windows 客户端传递选项时，可以仅用 push 指令简单地传递所需的选项。比如，给客户端传递一个 DNS 服务器 IP 地址，可以按以下方法做。

```
push "dhcp-option DNS 10.0.2.155"
```

这里设置 OpenVPN，让 Microsoft Windows 客户端推送 DNS 服务器地址 10.0.2.155 到它的 DHCP 选项。

在 Microsoft Windows 环境里，还可以推送其他多种选项，如表 14-1 所示。

表 14-1	DHCP 选项
选项	功能
DOMAIN 名称	设置客户端的 DNS 前缀
DNS 地址	设置 DNS 服务器地址，重复该选项可以设置多个 DNS 服务
WINS 地址	设置 WINS 服务器地址，重复该选项可以设置多个 WINS 服务器
NBDD 地址	设置 NBDD 服务器地址，重复该选项可以设置多个 NBDDS 服务器
NTP 地址	设置 NTP 服务器地址，重复该选项可以设置多个 NTP 服务器
DISABLE-NBT	禁用 TCP/IP 之上的 NetBIOS

在 Linux 和其他主机上，不可以用 `push` 指令直接设置上表的各个选项，而是需要配置 OpenVPN，让它在通道启用和关闭的时候运行一些脚本。这可以用 `up` 和 `down` 选项来配置。在 VPN 客户端的 `mobileclient.conf` 文件中添加以下选项。

```
up /etc/openvpn/tunnelup.sh
down /etc/openvpn/tunneldown.sh
```

每个选项指定在 VPN 通道启用和关闭时需要运行的脚本或命令。如果希望为客户端设置 DNS 配置，则要用到一个 `up` 脚本 `tunnelup.sh`，如下所示。

```
#!/bin/sh
mv /etc/resolv.conf /etc/resolv.conf.bak
echo "search example.org" > /etc/resolv.conf
echo "nameserver 10.0.2.155" >> /etc/resolv.conf
exit 0
```

接着，使用 `down` 脚本 `tunneldown.sh` 恢复之前的配置选项，如下所示。

```
#!/bin/sh
mv /etc/resolv.conf.bak /etc/resolv.conf
```

注：需要亲自把这些文件传给客户端，或者借助一个配置管理工具把它们传递给客户端。第 19 章介绍配置管理工具。

最后，可以强制让所有客户端的通信都经过 VPN。这常用于强制用户遵循一些原则或标准，或者让所有通信都经过一个代理或者病毒扫描器，以检查通信中是否含有病毒和恶意软件。

然而，强制所有通信都经过 VPN 通道还存在一些问题。最明显的问题是，通信经过通道到总公司，然后再转发到 Internet，其性能会受到影响。我们还需要为所有通道架设一个代理或者 NAT 重定向服务，因为任何协议——不只是 Web 通信协议——都需要一种方法来连接 Internet。

为了强制来自客户端的所有通信都经过 VPN 通道，对 `gateway` 主机上的配置文件 `mobile.conf` 添加以下指令。

```
push "redirect-gateway def1"
```

如果 VPN 通道建立在无线网络之上，并且所有客户端和服务端都处于同一个无线网络范围，则需要添加这一指令，如下所示。

```
push "redirect-gateway local def1"
```


可以看到上面的配置为该指令添加了 local 选项。

14.3 OpenVPN 故障排除

检查 OpenVPN 的故障需要考虑到 VPN 连接的所有方面：网络、防火墙和 OpenVPN 本身。OpenVPN 大量的日志（之前在本章我们已看过 log-append、status 和 verb 选项）可以帮助我们快速找出错误。另外，OpenVPN 的错误信息通常提供了关于实际问题的清楚而准确的提示。

除此之外，还需要检查网络是否连通，是否有合适的防火墙规则允许通信——既包括自己主机上的 IP 表规则，也包括任意中间网络设备的可能规则。还需要检查连接是否建立，也就是说防火墙规则是否允许 VPN 通道进行连接。最后，检查那些规则和路由是否存在，也就是说是否允许我们的通信经过 VPN 通道到达目的地址。

■注：第 6 章涉及网络和防火墙的故障查找和处理。

查找 OpenVPN 故障排查帮助文档的最好地方是 OpenVPN 网站 (<http://openvpn.net/>)。可以在那里找到相关文档，包括一个综合的 HOWTO 页面 (<http://openvpn.net/index.php/documentation/howto.html>)、一个 FAQ 页面 (<http://openvpn.net/index.php/documentation/faq.html>)。OpenVPN 的手册页面是 <http://openvpn.net/index.php/documentation/manuals/openvpn-20x-manpage.html>。我们还可以在 <http://openvpn.net/index.php/documentation/miscellaneous/mailling-lists.html> 上加入 OpenVPN 的邮件列表。对于更为复杂的应用，可以从 OpenVPN 的研发人员那里得到商业上的支持，还可以从 Markus Feilner 的《OpenVPN: Building and Integrating Virtual Private Networks》(Packt Publishing, 2006) 一书上寻求帮助。

14.4 小 结

本章带领读者经历了配置和管理 VPN 通道的整个过程，介绍了点对点 VPN 通道，例如总公司和远程分公司之间的通道。本章还介绍了如何使用 VPN 通道，以便移动用户可以安全地访问总公司资源或连接到其他地方。学习本章之后，读者应该掌握以下几点。

- 配置 VPN 通道。
- 为身份认证创建和配置证书。
- 使用 PAM 的身份认证方式。
- 配置防火墙的 IP 表，允许 VPN 通道通信。
- 配置网络和防火墙，允许用户穿越 VPN 通道并访问资源。
- 通过 OpenVPN 在客户端上配置网络选项。

下一章讨论诸如 e-mail 和日程表的团队协作工具。

第 15 章 协作服务



协作服务器 (collaboration server)，也叫群件 (groupware)，是消息服务和文档管理服务的集合。很多机构都要求人们能够便捷地沟通、安排会议、交流想法以及书写和共享文档。邮件、共享邮件夹、即时通信、共享日程表、共享任务表、共享文档以及共享工作区等被组合成一种产品，称为协作服务。现有的几个协作服务软件——包括免费的开源软件和非开源的商业软件——都被设计运行在 Linux 系统上。

本章将向读者介绍如何安装 Zimbra 服务器。选择该款服务器而不选择其他同类软件是基于下述关键理由。

- Zimbra 服务器易于安装和配置。
- Zimbra 服务器拥有完备的功能，可满足我们的所有需求。
- Zimbra 服务器的研发公司一如既往地支持开源社区。

没有理由不使用 Zimbra 的 Web 客户端或免费的桌面客户端来访问 Zimbra 服务器。Zimbra 减少了应付各种电子邮件客户端的麻烦，降低了防火墙的复杂程度，并简化了对网络与主机的管理。这意味着不再需要运行 IMAP 和 POP 服务，尽管在企业需要时这些服务也很容易配备。

此外，在这种网络环境下，可以认真考虑削减企业中的 Microsoft 和 Apple 主机的数目。因为许多客户端软件是基于 Web 的，所以可以在 Linux 桌面的主机上使用它们。基于桌面系统的 Linux 主机不仅适用于运行 Web 浏览器，还进一步降低了成本。然而，有些人仍然需要商业桌面系统来运行一些特定的应用程序。作为长远发展策略的一部分，可以考虑移除所有会把我们绑定到特定供应商的软件。

在企业中，可以在绝大多数主机上安装 Linux 系统并运行开源软件，这样每年节省成千上万的运营成本（这取决于企业的规模），而所有这些使运营成本比竞争对手更低的东西给我们带来了莫大的好处。

备选协作服务产品

以下是其他可用的协作产品列表，读者可以自行探讨。

- Horde: 初级的群件产品，整合了网页式电子邮件访问功能和日程表 (<http://www.horde.org/>)。
- Citadel: 完全免费的邮件和协作服务器。不如接下来的产品功能完整 (<http://www.citadel.org/doku.php>)。
- OpenXchange: 另一个开源的功能完备的协作服务器。商业机构需要付费才能获得授权 (<http://www.open-xchange.com/>)。
- Scalix: 另一个功能完备的开源协作服务器。是有限授权的社区版本 (community edition) (<http://www.scalix.com/community/>)。
- Novell Groupwise: 非开源的功能完备的协作服务，它运行在 Linux 上，由 Novell 公司出品。是商业授权产品 (<http://www.novell.com>)。
- Lotus Domino: 另一个非开源的协作服务，运行在 Linux 上，由 IBM 出品。是商业授权产品 (<http://www-01.ibm.com/software/lotus/>)。

15.1 Zimbra

正如之前提到的，选择 Zimbra 来展示协作服务是因为它完备的功能集合、开发公司对开源社区的支持以及该产品的无限制授权的社区版本 (community edition)。Zimbra 同样也是开源软件技术的典范，它以开源软件技术创造出优秀产品并仍忠实其开源初衷。Zimbra 公司提供可以添加到 Zimbra 服务器上的收费功能组件，这些功能包括发送电子邮件到掌上电脑和移动设备，以及商业支持协议。

Zimbra 本身可以安装在多种不同的工作环境中。它可以使用外部认证服务，如 OpenLDAP。我们可以把多个电子邮件存储服务器分布在网络中，以利于快速访问位于不同地理区域的服务器。Zimbra 支持社区成员创建的被称为 Zimlet 的附加组件，某些 Zimlet 组件甚至允许我们在 Zimbra 提供的管理控制台 (Administration Console) 下管理 Samba 域。

■注：Zimlet 是社区开发的附加软件，它们可以为 Zimbra 服务器增加新的功能。如果我们需要某种功能或组合应用，可以创建自己的 Zimlet，还可以把自己创建的 Zimlet 发布到社区供别人使用。

图 15-1 展示了一个复杂的 Zimbra 解决方案，该图基于出现在 ZCS Administrator's Guide 中的插图。

如图 15-1 所示，我们的电子邮件系统可以变得非常复杂。在图 15-1 里，可以看到该电子邮件系统在设计方案里设置了许多安全构件和冗余构件。进入公司的电子邮件将进入位于防火墙之后的边缘 MTA (邮件传送代理)，在那里，可以远离内部邮件服务器做一些初步的垃圾邮件和病毒检查。一个负载均衡器 (load balancer) 把用户负载分配给众多 Zimbra 服务器，用户在某个 Zimbra 服务器上通过验证后使用 LDAP 服务，然后电子邮件被真正发送到该 Zimbra 服务器，于是用户就可以登录该 Zimbra 服务器并获取电子邮件。

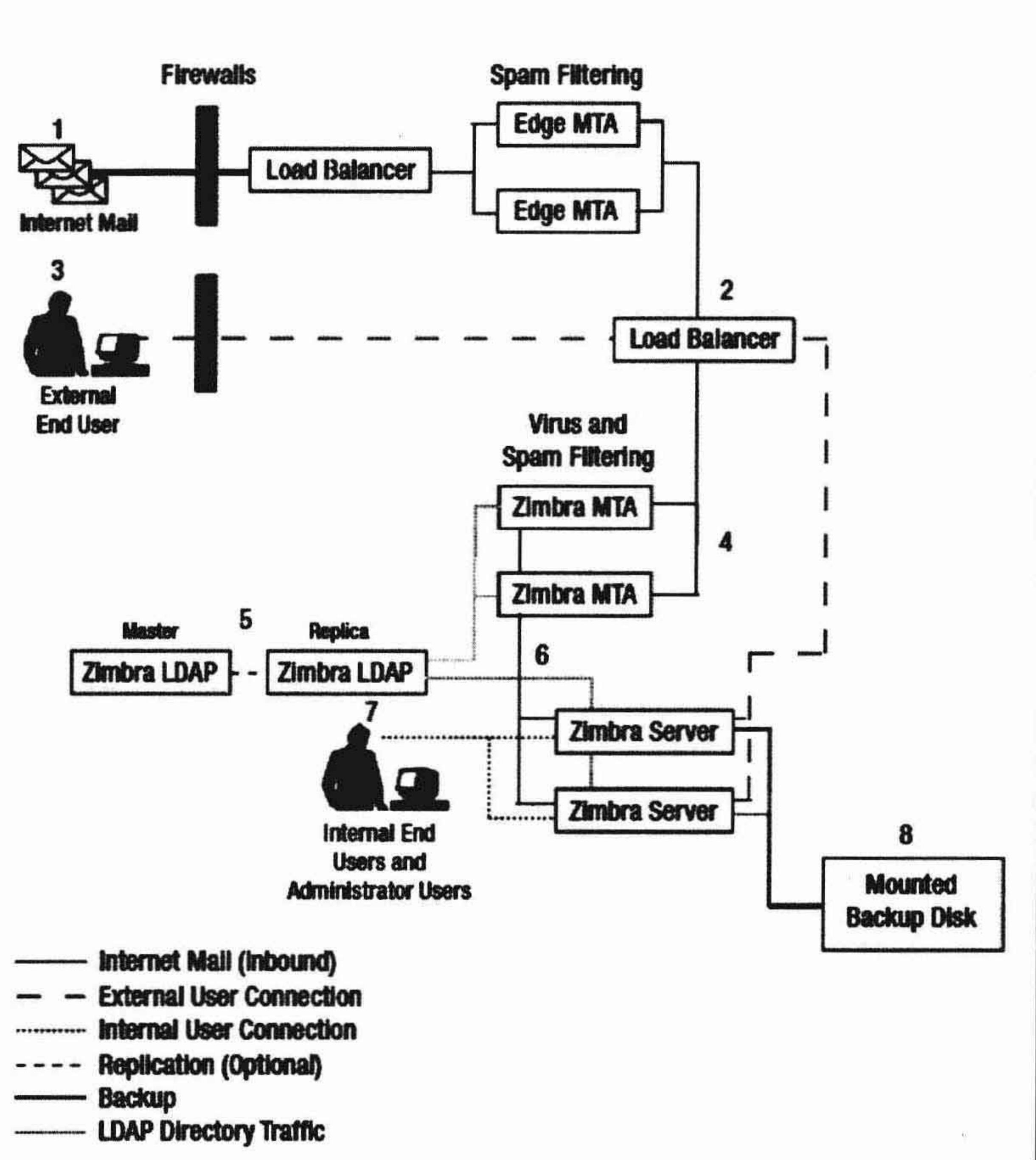


图 15-1 一个复杂的 Zimbra 解决方案

下面将引导读者做一次常用的标准 Zimbra 安装。假设读者记得第 6 章所描述的办公场景，在别的地区还有数个子子公司通过 VPN（虚拟专用网络）连接主公司。本节将要介绍的 Zimbra 安装适用于上述情况，无需对公司网络做额外改动或添加硬件设备。而且，较之这里介绍的应用场景，Zimbra 还可用于更为复杂的情况。

■注：第 14 章讲述如何建立自己的 VPN 连接。

在一台性能优越的主机上安装的单个 Zimbra 服务器可以应付 500 到 1000 个用户。一旦公司的用户数目达到此规模上限，明智的做法是削减用户数目和进行负载均衡，因为 500 个不能收发电子邮件的用户将导致业务风险。

图 15-2 展示了本地的公司网络。在这个场景中，有一台服务器提供各种服务，包括 DHCP 服务、DNS 服务和邮件服务。下面准备把 Zimbra 安装在主机 headoffice.example.com 上。

我们只用一个 Zimbra 服务器来提供所有的邮件服务。在本例中，替换掉在第 10 章所创建的邮件服务器。Zimbra 使用的基本 MTA 是 Postfix，所以读者仍然可以参考已有的 Postfix 知识来理解 Zimbra 是如何工作的。

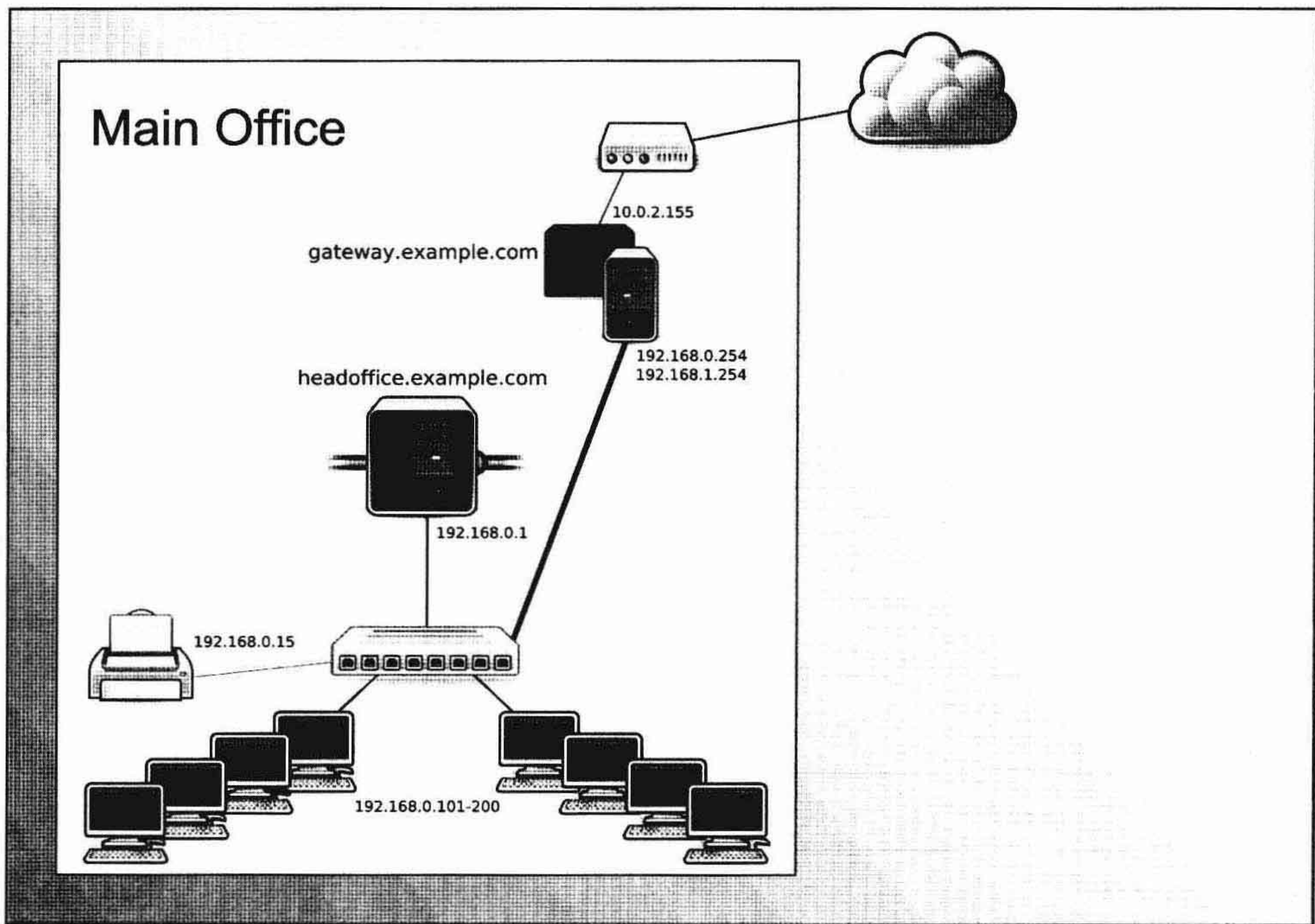


图 15-2 我们的网络计划

15.2 安装 Zimbra

Zimbra 的安装过程很容易完成，而且无论选择的主机是 Ubuntu 系统还是 Red Had 系统，除了稍后提到的少许不同之外，它们的 Zimbra 安装过程是一样的。Zimbra 的安装程序是基于文本的控制台程序，它会给用户一系列关于想要安装哪些服务的询问。安装过程大约花费 20 分钟。

15.2.1 安装前提

在安装 Zimbra 之前，系统必须满足某些前提条件。Zimbra 安装在 /opt 目录下，/opt 目录必须有超过 5GB 的剩余空间，而 /tmp 目录必须有至少 100MB 的剩余空间。另外，还必须在主机上安装某些特定的软件。

Ubuntu 主机的某些安装前提条件与 Red Had 主机不同。在 Ubuntu 主机上，确保系统已安装下列软件（可以跳过已经安装好的软件包）。

```
$ sudo aptitude install libidn11 libpcrc3 libgmp3c2 libexpat1 libstdc++6  
libstdc++5 libltdl3 perl
```

Red Had 主机必须安装下列软件。

```
$ sudo yum install sudo libidn fetchmail gmp compat-libstdc++ libstdc++  
libtool-ltdl perl
```


需要知道的重要一点是，这些前提条件可能会随着时间而改变。安装过程本身会检查所有前提条件，如果某些条件不满足，会要求我们安装缺失的软件包，然后才能继续安装。这里可以使用 `aptitude` 或 `yum` 命令搜索缺失的软件包。

接下来要做的是关闭当前可能会与安装服务发生冲突的所有服务。必须关闭正在主机上运行的任何 `SMTP`、`IMAP`、`POP3`、`HTTP`、`HTTPS` 和 `LDAP` 服务，并确保在主机重启后这些服务不会重新运行。

在 `Ubuntu` 主机上，输入下列命令停止这些服务。

```
$ sudo /usr/sbin/invoke-rc.d dovecot stop
$ sudo /usr/sbin/invoke-rc.d postfix stop
$ sudo /usr/sbin/invoke-rc.d apache2 stop
$ sudo /usr/sbin/invoke-rc.d mysql stop
$ sudo /usr/sbin/invoke-rc.d slapd stop
```

还要禁止它们在系统重启之后再次启动。

```
$ sudo /usr/sbin/update-rc.d -f dovecot remove
$ sudo /usr/sbin/update-rc.d -f postfix remove
$ sudo /usr/sbin/update-rc.d -f apache2 remove
$ sudo /usr/sbin/update-rc.d -f mysql remove
$ sudo /usr/sbin/update-rc.d -f slapd remove
```

对于 `Red Had` 主机，用以下命令停止这些服务。

```
$ sudo /sbin/service dovecot stop
$ sudo /sbin/service postfix stop
$ sudo /sbin/service httpd stop
$ sudo /sbin/service mysqld stop
$ sudo /sbin/service slapd stop
```

同样禁止它们在系统重启之后再次启动。

```
$ sudo /sbin/chkconfig dovecot off
$ sudo /sbin/chkconfig postfix off
$ sudo /sbin/chkconfig httpd off
$ sudo /sbin/chkconfig mysqld off
$ sudo /sbin/chkconfig slapd off
```

有些服务也许并未安装在主机上，这取决于已安装的软件。请用第 6 章讨论过的 `netstat` 命令检查这些服务是否已经停止。现在，可以继续 `Zimbra` 的安装过程，下载 `Zimbra` 软件包并为安装做好准备。

15.2.2 下载和主机前期准备

可以从 `Zimbra` 网站获得 `Zimbra` 服务器软件包。我们可以用浏览器下载 `Zimbra` 压缩包，或者复制下载链接并用 `wget` 命令下载该软件包。所有下载包可以通过此链接获得：<http://www.zimbra.com/community/downloads.html>。

在本章的具体例子中，我们在 `Ubuntu` 主机上安装 `Zimbra`。初始安装之后，无论在 `Red Had` 主机上还是在 `Ubuntu` 主机上，`Zimbra` 程序的执行都一样。在 `Zimbra` 的下载网站上可以发现，

对应于不同的 Ubuntu 发行版本，有不同的 Zimbra 软件包可供下载。我们选择一个和 Ubuntu 8.04 LTS (32-bit x86)相适应的软件包，并用 `wget` 命令下载所选文件。以下便是一个从特定 URL 下载文件的例子。

```
$ wget http://h.yimg.com/lo/downloads/version/file
```

所下载的文件 `file` 在一个 `tar` 压缩包里，所以需要用到 `tar` 程序，加上 `gzip` 标志 “-z”，解压缩该下载包。

```
$ tar xzf zcs-5.0.13_GA_2791.UBUNTU8.20090206174622.tgz
```

新建的目录叫做 `zcs-5.0.13_GA_2791.UBUNTU8.20090206174622`。现在切换到该目录，并列出该目录下的文件和文件夹。

```
$ cd zcs-5.0.13_GA_2791.UBUNTU8.20090206174622/
```

```
$ ls
```

```
bin data docs install.sh
packages readme_binary_en_US.txt
readme_source_en_US.txt README.txt util
```

该目录下有一个 `docs` 子目录和一个 `package` 目录。`docs` 目录里面含有几个 PDF 帮助文档（这些 PDF 文档也可以从 Zimbra 网站下载）；`packages` 目录含有各个软件包。下面用 `install.sh` 脚本来安装 `packages` 中的软件包。

在执行 `install.sh` 脚本之前，需要编辑 `/etc/hosts` 文件，确保该文件中只有如下几行。

```
127.0.0.1 localhost.localdomain localhost
192.168.0.1 mail.example.com
```

该 `hosts` 文件用于匹配主机名和 IP 地址。做上述改动是因为 Zimbra 把 MySQL 用户的权限设置为只允许从回环地址（127.0.0.1）发起连接。如果不按照上述方法进行设置，Zimbra 将会给出一个让人费解的安装错误，即要求修改 `hosts` 文件。

读者还会看到我们把主机名 `mail.example.com` 添加到 `hosts` 文件，并指定其 IP 地址为 192.168.0.1。该主机 IP 已经在 DNS 域文件（zone file）里配置过了，其对应的主机名为 `headoffice.example.com`。另外，还必须确保 `mail.example.com` 也被解析成 IP 地址 192.168.0.1。

我们还要在 DNS 服务中创建 MX 记录，并为邮件服务器主机设置一个 CNAME 记录。如果当前使用的是在第 9 章安装的 BIND DNS 服务器，必须确保域的 MX 记录指向将来用来做邮件服务器的主机。在 `example.com` 的域文件里，需要添加以下内容。

```
mail IN      CNAME      headoffice.example.com.
example.com. IN MX      10 mail
```

改动后的配置为指向主机 `headoffice.example.com` 的 `mail` 记录添加了一个 CNAME 记录。现在有了一个对应于 `headoffice.example.com` 的 A 记录，该 A 记录指向 IP 地址 192.168.0.1。这里的 CNAME 记录对主机名取别名以区别于其他记录。然后用该 CNAME 记录在域文件中设置一个指向 `mail.example.com`、优先级为 10 的 MX 记录。MX 记录指明一个域的邮件交换主机，其优先级允许设置多个 MX 记录，并优先选择优先级小的记录。

■注：可以在网站 “http://www.zimbra.com/docs/ne/latest/single_server_install/” 上查看 Zimbra 的速学安装教程。

15.2.3 安装 Zimbra

安装 Zimbra 相对容易。在完成前提准备之后，需要执行 `install.sh` 脚本。该脚本启动基于文本的安装过程，安装过程需要我们回应一些询问。安装 Zimbra 需要 `root` 用户权限，因此必须使用 `sudo` 命令执行该脚本。

```
$ sudo ./install.sh
Operations logged to /tmp/install.log.5430
Checking for existing installation...
  zimbra-ldap...NOT FOUND
  zimbra-logger...NOT FOUND
  zimbra-mta...NOT FOUND
  zimbra-snmp...NOT FOUND
  zimbra-store...NOT FOUND
  zimbra-apache...NOT FOUND
  zimbra-spell...NOT FOUND
  zimbra-proxy...NOT FOUND
  zimbra-archiving...NOT FOUND
  zimbra-convert...NOT FOUND
  zimbra-cluster...NOT FOUND
  zimbra-core...NOT FOUND
```

安装程序启动时，它首先检查主机是否已经安装过任何 Zimbra 组件。本章的安装是一次全新安装，所以不出所料这里会看到每一个列出来的组件都是 `NOT FOUND`。

接着，安装程序检查之前已经安装好的依赖软件。如果漏掉任何一项，它都会被标记成 `MISSING` 出现在这里，我们必须安装好缺失的依赖软件才能完成 Zimbra 安装。

```
Checking for prerequisites...
  FOUND: NPTL
  FOUND: sudo-1.6.9p10-1ubuntu3.4
  FOUND: libidn11-1.1-1
  FOUND: libpcre3-7.4-1ubuntu2.1
  FOUND: libgmp3c2-2:4.2.2+dfsg-1ubuntu2
  FOUND: libexpat1-2.0.1-0ubuntu1
  FOUND: libstdc++6-4.2.4-1ubuntu3
  FOUND: libstdc++5
  FOUND: libltdl3-1.5.26-1ubuntu1
Checking for suggested prerequisites...
  FOUND: perl-5.8.8
```

现在，该选择安装哪些 Zimbra 组件了。可以在不同的主机上配置不同的 Zimbra 组件。例如，可以把认证服务安装在数台主机上，而把邮件存储服务分开安装到本地其他的服务器或者不同域的服务器上。

■注：要获得关于高级 Zimbra 安装的信息，请查阅 *ZCS Administrator's Guide* 上的产品概述见“http://www.zimbra.com/docs/os/latest/administration_guide/2_Overview%20System%20Architecture.3.1.html#1101622”。

本章准备安装不带 `zimbra-proxy` 组件的基本 Zimbra 服务器。`zimbra-proxy` 服务允许用户

使用代理，所谓的“使用代理”就是在其他主机上运行这些服务，如 IMAP 和 POP。由于本章并不打算使用 IMAP 和 POP，因此不安装 `zimbra-proxy` 组件。

下面是必须要安装的软件包：

```
Select the packages to install
Install zimbra-ldap [Y]
```

`zimbra-ldap` 软件包是一种认证服务，它管理 OpenLDAP 服务器上的用户配置详情。高级的配置可能需要多个 LDAP 服务器，其中一个服务器作为主服务器，其余作为备份服务器。

```
Install zimbra-logger [Y]
```

这里选择安装 Zimbra 日志服务，该服务提供一些日志工具，用于收集、报告系统记录以及跟踪报文。要想使用管理控制台（Administration Console）的信息统计功能，必须安装这个服务。

■注：第 18 章对系统记录和日志做更多讨论。

```
Install zimbra-mta [Y]
```

`zimbra-mta` 安装一个 Postfix 邮件服务器用于收发邮件（该邮件服务器还具有反病毒和反垃圾邮件功能）。

```
Install zimbra-snmp [Y]
```

SNMP 是 Simple Network Management Protocol 的简写，它可用于监控网络中的主机。Zimbra（`zimbra-ldap`、`zimbra-mta` 和 `zimbra-core`）也使用 SNMP 监控系统记录和发送 SNMP 陷阱（SNMP trap）。

■注：第 17 章讨论 SNMP。读者可以从以下网站读到更多关于 SNMP 的内容，见“http://en.wikipedia.org/wiki/Simple_Network_Management_Protocol”。

```
Install zimbra-store [Y]
```

`zimbra-store` 就是邮件存储服务。在高级 Zimbra 配置里，可以在网络中安装多个 `zimbra-store` 服务器。这些服务器为用户保存报文和用户数据。每当创建一个用户，该用户就会被分配给一个邮件存储服务器，当该用户登录邮箱时，他就会从 Web 接口或 POP/IMAP 服务器连接到之前分配的邮件存储服务器。

```
Install zimbra-apache [Y]
```

`zimbra-apache` 安装 Apache 服务器，从而提供 Web 接口服务。

```
Install zimbra-spell [Y]
```

`zimbra-spell` 组件被 Web 客户端用于拼写检查。

```
Install zimbra-proxy [N]
```

如果选择安装 `zimbra-proxy` 服务，就可以在网络中部署代理 IMAP 和 POP 主机。这些代理主机把客户端的 IMAP/POP 连接导向合适的邮件存储服务器。


```
checking space for zimbra-store
```

```
Installing:
```

```
zimbra-core
zimbra-ldap
zimbra-logger
zimbra-mta
zimbra-snmp
zimbra-store
zimbra-apache
zimbra-spell
```

```
The system will be modified. Continue? [N] Y
```

```
Installing packages
```

```
zimbra-core.....zimbra-core_5.0.13_GA_2791.UBUNTU8_i386.deb...done
zimbra-ldap.....zimbra-ldap_5.0.13_GA_2791.UBUNTU8_i386.deb...done
zimbra-logger.....zimbra-logger_5.0.13_GA_2791.UBUNTU8_i386.deb...done
zimbra-mta.....zimbra-mta_5.0.13_GA_2791.UBUNTU8_i386.deb...done
zimbra-snmp.....zimbra-snmp_5.0.13_GA_2791.UBUNTU8_i386.deb...done
zimbra-store.....zimbra-store_5.0.13_GA_2791.UBUNTU8_i386.deb...done
zimbra-apache.....zimbra-apache_5.0.13_GA_2791.UBUNTU8_i386.deb...done
zimbra-spell.....zimbra-spell_5.0.13_GA_2791.UBUNTU8_i386.deb...done
```

```
Operations logged to /tmp/zmsetup.03102009-215512.log
```

现在，已经把这些软件包安装在主机上了。接下来，Zimbra 安装程序将进行一些安装后的配置。

```
Setting defaults...
```

```
Interface: 127.0.0.1
```

```
Interface: 192.168.0.1
```

设置主机的网络接口地址。

```
DNS ERROR - none of the MX records for mail.example.com
```

```
resolve to this host
```

```
Change domain name? [Yes] Yes
```

```
Create Domain: [mail.example.com] example.com
```

```
MX: mail.example.com (192.168.0.1)
```

```
Interface: 127.0.0.1
```

```
Interface: 192.168.0.1
```

```
done.
```

```
Checking for port conflicts
```

如果系统中有邮件服务或 Web 服务正在监听标准端口——比如 80、443、110、25 等——那么需要停止这些已存在的邮件服务或者 Web 服务，并让它们不再运行。

15.2.4 Zimbra 安装后续的配置菜单

选择好需要安装的 Zimbra 组件之后会看到一个配置菜单，该菜单允许配置 Zimbra 服务器的各个方面。这是设置密码的地方，也是配置 Zimbra 组件的地方。为了完成 Zimbra 安装，使用这个选项驱动的配置菜单，设置 Admin 用户密码，该设置项由*****号标出。

Main menu

```

1) Common Configuration:
2) zimbra-ldap: Enabled
3) zimbra-store: Enabled
   +Create Admin User: yes
   +Admin user to create: admin@example.com
****
   +Admin Password UNSET
   +Enable automated spam training: yes
   +Spam training user: spam.syuqmdsha@example.com
   +Non-spam(Ham) training user: ham.qiomilor@example.com
   +Global Documents Account: wiki@example.com
   +SMTP host: mail.example.com
   +Web server HTTP port: 80
   +Web server HTTPS port: 443
   +Web server mode: http
   +IMAP server port: 143
   +IMAP server SSL port: 993
   +POP server port: 110
   +POP server SSL port: 995
   +Use spell check server: yes
   +Spell server URL: http://mail.example.com:7780/aspell.php
   +Configure store for use with reverse mail proxy: FALSE
   +Configure store for use with reverse Web proxy: FALSE

4) zimbra-mta: Enabled
5) zimbra-snmp: Enabled
6) zimbra-logger: Enabled
7) zimbra-spell: Enabled
8) Default Class of Service Configuration:
r) Start servers after configuration yes
s) Save config to file
x) Expand menu
q) Quit

```

Address unconfigured (**) items (? - help) 3

正如读者所看到的，第 3 个菜单项 `zimbra-store` 下面有想要使用的 Admin Password 设置项，所以选择该菜单项。这里显示的子菜单允许我们选择其第 4 项子菜单并按回车，然后进入 Admin Password 选项。

Store configuration

```

1) Status: Enabled
2) Create Admin User: yes
3) Admin user to create: admin@example.com
** 4) Admin Password UNSET
5) Enable automated spam training: yes
6) Spam training user: spam.syuqmdsha@example.com
7) Non-spam(Ham) training user: ham.qiomilor@example.com
8) Global Documents Account: wiki@example.com
9) SMTP host: mail.example.com
10) Web server HTTP port: 80
11) Web server HTTPS port: 443
12) Web server mode: http
13) IMAP server port: 143
14) IMAP server SSL port: 993
15) POP server port: 110

```


- 16) POP server SSL port: 995
- 17) Use spell check server: yes
- 18) Spell server URL: http://mail.example.com:7780/aspell.php
- 19) Configure store for use with reverse mail proxy: FALSE
- 20) Configure store for use with reverse Web proxy: FALSE

Select, or 'r' for previous menu [r] 4

正如读者所看到的，当进入某个子菜单，可以输入 r 键返回上级菜单，也可以输入当前显示的序号选择某项菜单。在此，选择第 4 项准备为 Admin 用户添加一个密码。这时会自动生成一个密码，如下行的中括号包括的字符串所示。

Password for admin@example.com (min 6 characters): [0aPFi_01Ta] somepassword123

这里也可以自己设定密码。

输入 r 返回主菜单。当前，仅需设置 Admin 用户的密码，当然也可以在管理控制台上设置其他选项。

- Main menu
- 1) Common Configuration:
 - 2) zimbra-ldap: Enabled
 - 3) zimbra-store: Enabled
 - 4) zimbra-mta: Enabled
 - 5) zimbra-snmp: Enabled
 - 6) zimbra-logger: Enabled
 - 7) zimbra-spell: Enabled
 - 8) Default Class of Service Configuration:
 - r) Start servers after configuration yes
 - s) Save config to file
 - x) Expand menu
 - q) Quit

*** CONFIGURATION COMPLETE - press 'a' to apply
Select from menu, or press 'a' to apply config (? - help) a

为 Admin 用户设置好新密码后，就准备提交对 Zimbra 配置所做的修改。为此，输入 a，然后回车。下一步，安装程序询问我们是否把配置信息保存到一个文件，然后生成 SSL 认证，最后完成其余安装步骤。

Save configuration data to a file? [Yes]
Save config in file: [/opt/zimbra/config.11384]
Saving config in /opt/zimbra/config.11384...done.
The system will be modified - continue? [No] Yes
Operations logged to /tmp/zmsetup.03102009-215512.log
Setting local config values...done.
Setting up CA...done.
Deploying CA to /opt/zimbra/conf/ca ...done.
Creating SSL certificate...done.
Installing mailboxd SSL certificates...done.
Initializing ldap...done.
Setting replication password...done.
Setting Postfix password...done.
Setting amavis password...done.
Setting nginx password...done.
Saving CA in ldap ...done.


```
Creating server entry for mail.example.com...done.
Saving SSL Certificate in ldap ...done.
Setting spell check URL...done.
Setting service ports on mail.example.com...done.
Adding mail.example.com to zimbraMailHostPool in default COS...done.
Installing skins...
    steel
    beach
    sky
    zmail
    yahoo
    lavender
    bones
    bare
    lemongrass
    sand
    hotrod
    waves
done.
Setting zimbraFeatureIMEnabled=FALSE...done.
Setting zimbraFeatureTasksEnabled=TRUE...done.
Setting zimbraFeatureBriefcasesEnabled=TRUE...done.
Setting zimbraFeatureNotebookEnabled=TRUE...done.
Setting MTA auth host...done.
Setting TimeZone Preference...done.
Creating domain example.com...done.
Creating user admin@example.com...done.
Creating postmaster alias...done.
Creating user wiki@example.com...done.
Creating user spam.syuqmdsha@example.com...done.
Creating user ham.qiomilor@example.com...done.
Setting spam training accounts...done.
Initializing store sql database...done.
Setting zimbraSmtphostname for mail.example.com...done.
Initializing logger sql database...done.
Initializing mta config...done.
Configuring SNMP...done.
Setting services on mail.example.com...done.
Setting up syslog.conf...done.
```

You have the option of notifying Zimbra of your installation.
This helps us to track the uptake of the Zimbra Collaboration Suite.
The only information that will be transmitted is:

```
    The VERSION of zcs installed (5.0.13_GA_2791_UBUNTU8)
    The ADMIN EMAIL ADDRESS created (admin@example.com)
```

```
Notify Zimbra of your installation? [Yes] Yes
Notifying Zimbra of installation via http://www.zimbra.com/cgi-bin/notify.cgi? ➡
VER=5.0.13_GA_2791_UBUNTU8&MAIL=admin@example.com
```

Notification complete

```
Starting servers...done.
Checking for deprecated zimlets...done.
Installing common zimlets...
    com_zimbra_date...done.
```



```
com_zimbra_cert_manager...done.
com_zimbra_ymemoticons...done.
com_zimbra_phone...done.
com_zimbra_url...done.
com_zimbra_bulkprovision...done.
com_zimbra_local...done.
com_zimbra_email...done.
Finished installing common zimlets.
Initializing Documents...done.
Restarting mailboxd...done.
Setting up zimbra crontab...done.

Moving /tmp/zmsetup.03102009-215512.log to /opt/zimbra/log

Configuration complete - press return to exit
```

现在已经完成了 Zimbra 的安装，需要启动 Zimbra 服务。可以采用两种方法启动服务：使用 Zimbra 脚本启动服务（可以在 `/etc/init.d` 目录下找到该脚本）；或者输入 `su` 命令切换到 `zimbra` 用户，用 `zmcontrol` 命令停止和启动 Zimbra 服务。

对 Red Hat 主机，使用命令：

```
$ sudo /sbin/service zimbra stop|start|restart
```

对 Ubuntu 主机，使用命令：

```
$ sudo /usr/sbin/invoke-rc.d zimbra stop|start|restart
```

■注：在第 5 章，我们讲述过如何启动、停止服务，以及如何把服务添加到主机。

如果想用 `zmcontrol` 命令启动和停止 Zimbra 服务，则输入以下命令：

```
$ sudo su - zimbra
$ zmcontrol start|stop
```

用户 `zimbra` 可用于管理 Zimbra 服务的其他方面。关于用户 `zimbra` 的可用命令，可以从以下网页获得更多信息：http://www.zimbra.com/docs/os/latest/administration_guide/A_app-command-line.13.1.html。

15.2.5 更改防火墙

安装 Zimbra 要求的防火墙设置很简单，只要求用户能够访问主机进行邮件收发和通过 HTTP/HTTPS 获取邮件。此外，Zimbra 客户端和发送邮件到移动设备的客户端能够通过 HTTPS 发送通信数据。一个特别的设置是打开端口 7071 供管理控制台使用，但本端口被限制只允许通过本地网络与 Zimbra 主机连接。

■注：出于安全方面的考虑，建议不要通过 Internet 直接连接 Zimbra 的管理控制台。

```
$ sudo /sbin/iptables -I Firewall-eth0-INPUT rule_number -p tcp -m state --state NEW -m tcp --dport 25 -j ACCEPT
```



```
$ sudo /sbin/iptables -I Firewall-eth0-INPUT rule_number -p tcp -m state -state➤
-m tcp NEW --dport 80 -j ACCEPT
$ sudo /sbin/iptables -I Firewall-eth0-INPUT rule_number -p tcp -m state -state➤
-m tcp NEW --dport 443 -j ACCEPT
$ sudo /sbin/iptables -I Firewall-eth0-INPUT rule_number -s 192.168.0.0/24 -p tcp➤
-m state -state -m tcp NEW --dport 7071 -j ACCEPT
```

第 6 章介绍过如何设置防火墙。在这里，用命令 `iptables` 开启端口 25、80、443 和 7071。另外，`Firewall-eth0-INPUT` 和 `rule_number` 依赖于所使用的主机。

如果决定开放 Zimbra 主机的 POP 和 IMAP 访问，那么必须允许这些服务穿过防火墙。

```
$ sudo /sbin/iptables -I Firewall-eth0-INPUT rule_number -p tcp -m state -state➤
-m tcp NEW --dport 110 -j ACCEPT
$ sudo /sbin/iptables -I Firewall-eth0-INPUT rule_number -p tcp -m state -state➤
-m tcp NEW --dport 993 -j ACCEPT
$ sudo /sbin/iptables -I Firewall-eth0-INPUT rule_number -p tcp -m state -state➤
-m tcp NEW --dport 143 -j ACCEPT
$ sudo /sbin/iptables -I Firewall-eth0-INPUT rule_number -p tcp -m state -state➤
-m tcp NEW --dport 995 -j ACCEPT
```

接下来介绍如何使用管理控制台管理用户和设置服务。

15.2.6 Zimbra 管理控制台

在本节，我们登录管理控制台，然后开始设置剩余的 Zimbra 服务。可以使用 HTTPS 协议和 7071 端口在浏览器中打开 `https://mail.example.com:7071`，找到 Zimbra 管理控制台。

图 15-3 显示管理控制台的登录界面。在登录界面中必须提供用户名（在本例中，用户名为 `admin@example.com`）和先前所设密码，从而获得管理控制台的访问权。一旦登录成功，将会看到管理控制台的面板，如图 15-4 所示。

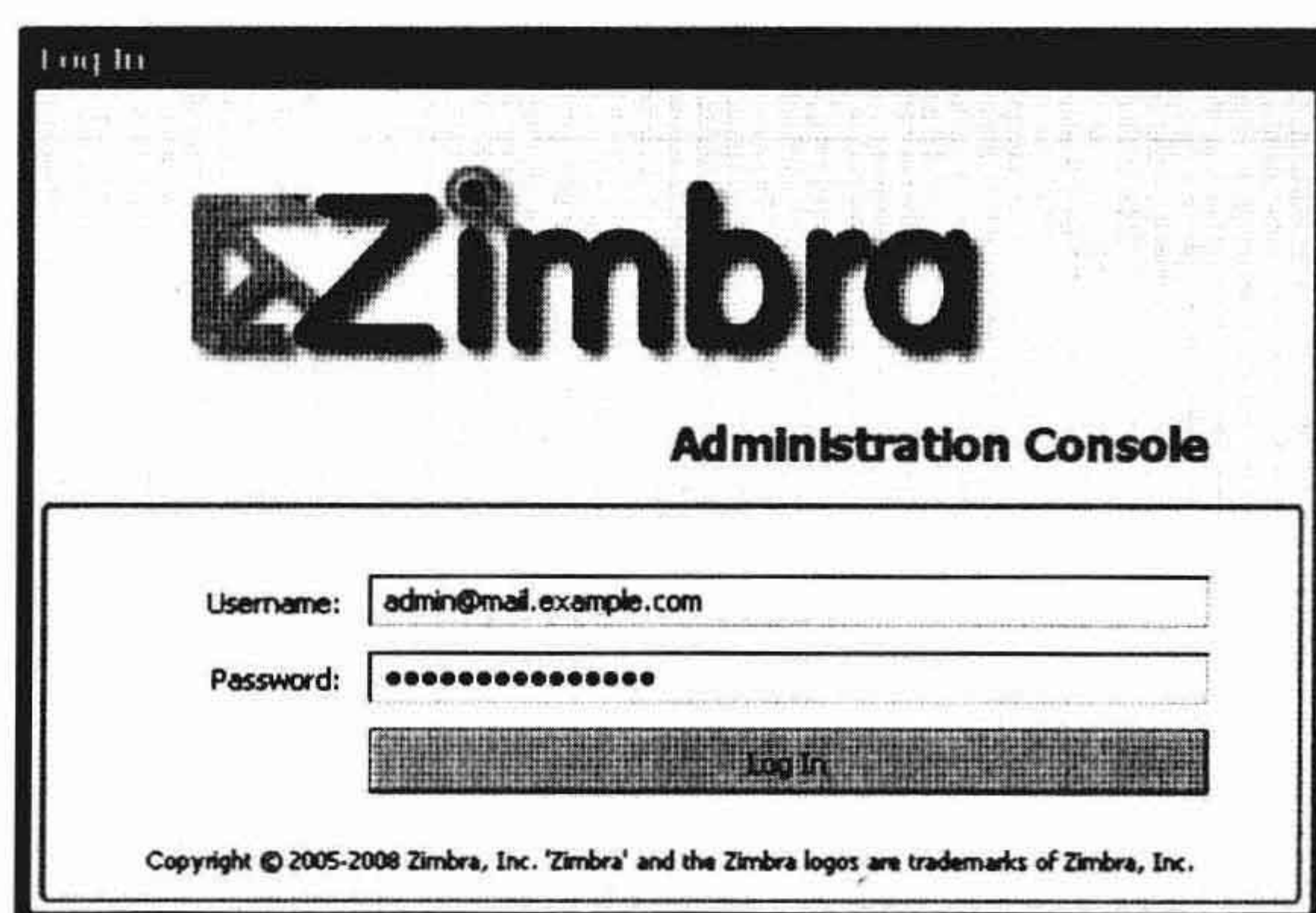


图 15-3 Zimbra 管理控制台登录界面

正如读者在图 15-4 中所看到的，呈现在我们面前的是首次登录后的服务器状态信息。注意，主界面中的所有服务都是活动的。左边界面显示的是服务器的各个部分，我们可以配置或查看这些部分。每当选中左边选项里的某一项，就会有一个新的标签页在主界面里打开。可以打开多个标签页，然后选择当前所要操作的某个标签页。Zimbra 使用一种叫做 Class of Service (CoS) 的对象，它可以用来设置 Zimbra 服务的默认设置项。默认设置项的范围包括密

码、邮件主机和界面主题等。下面准备创建自己的 CoS 用于域 example.com。

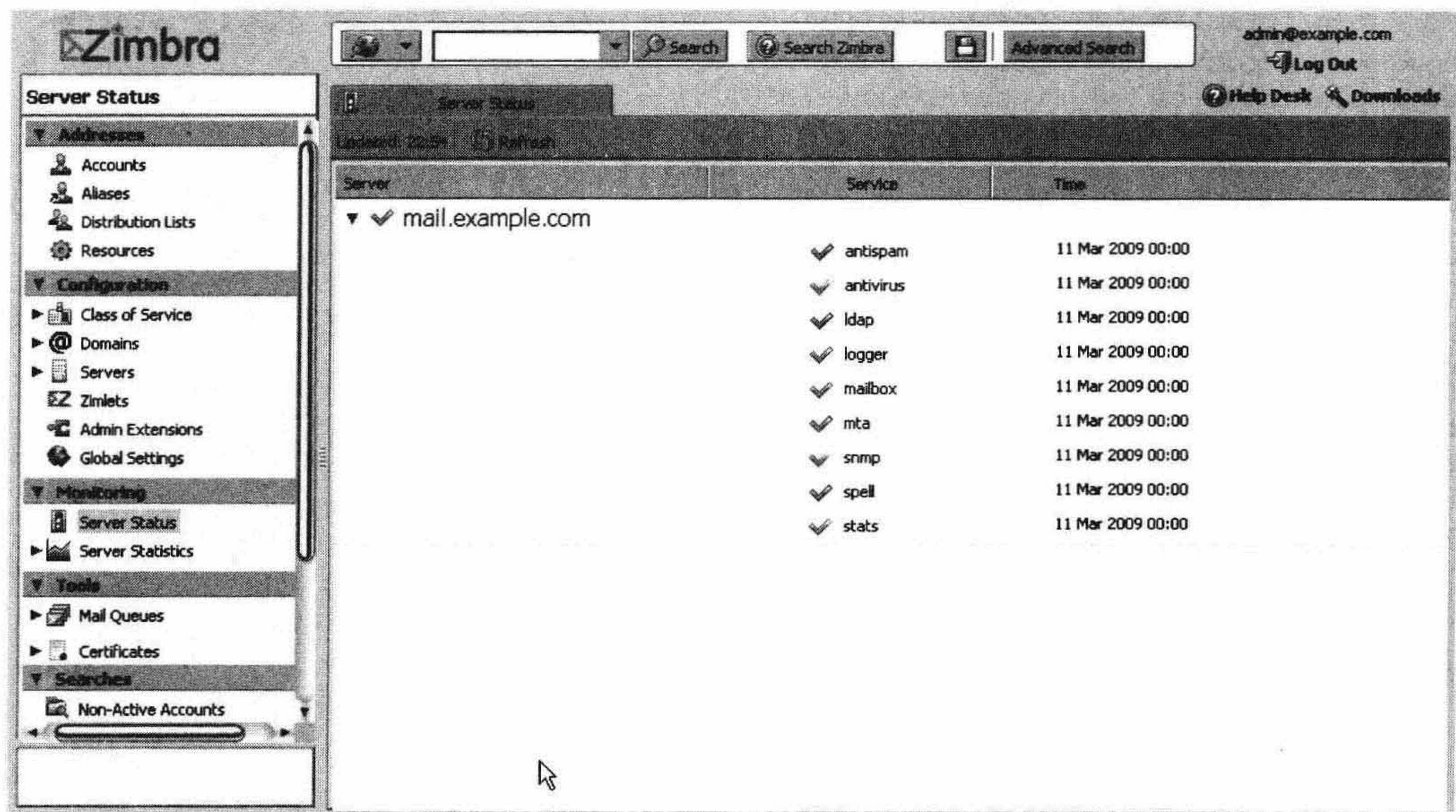


图 15-4 Zimbra 管理控制台

15.2.7 创建 Class of Service

在图 15-5 中，可以看到已经有了一个默认的 CoS，我们准备复制它。

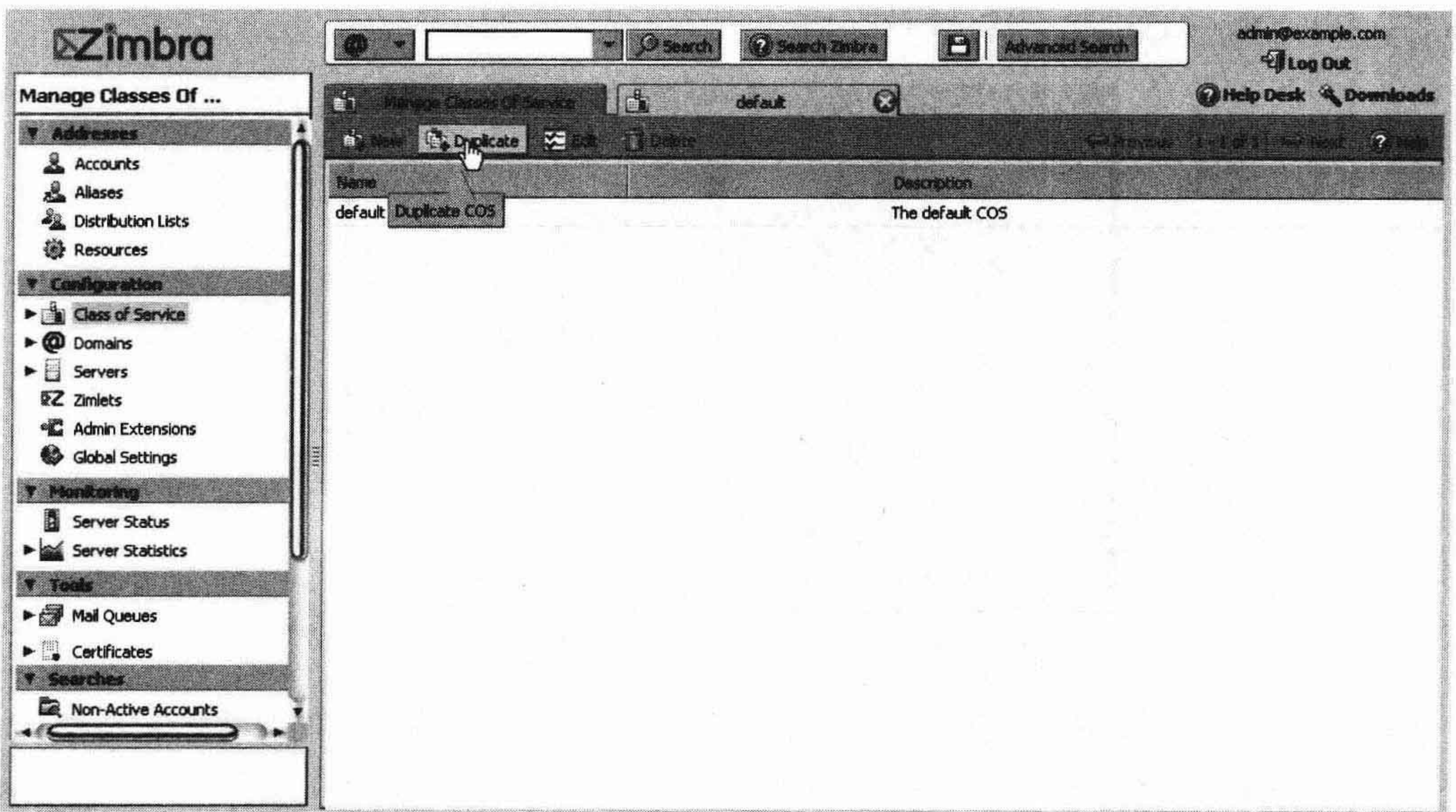


图 15-5 复制默认 CoS

复制默认的 CoS 既可以复用它的部分设置，又可以更改所需要的部分。如图 15-6 所示，

它显示如何添加一些描述 CoS 的概要信息。

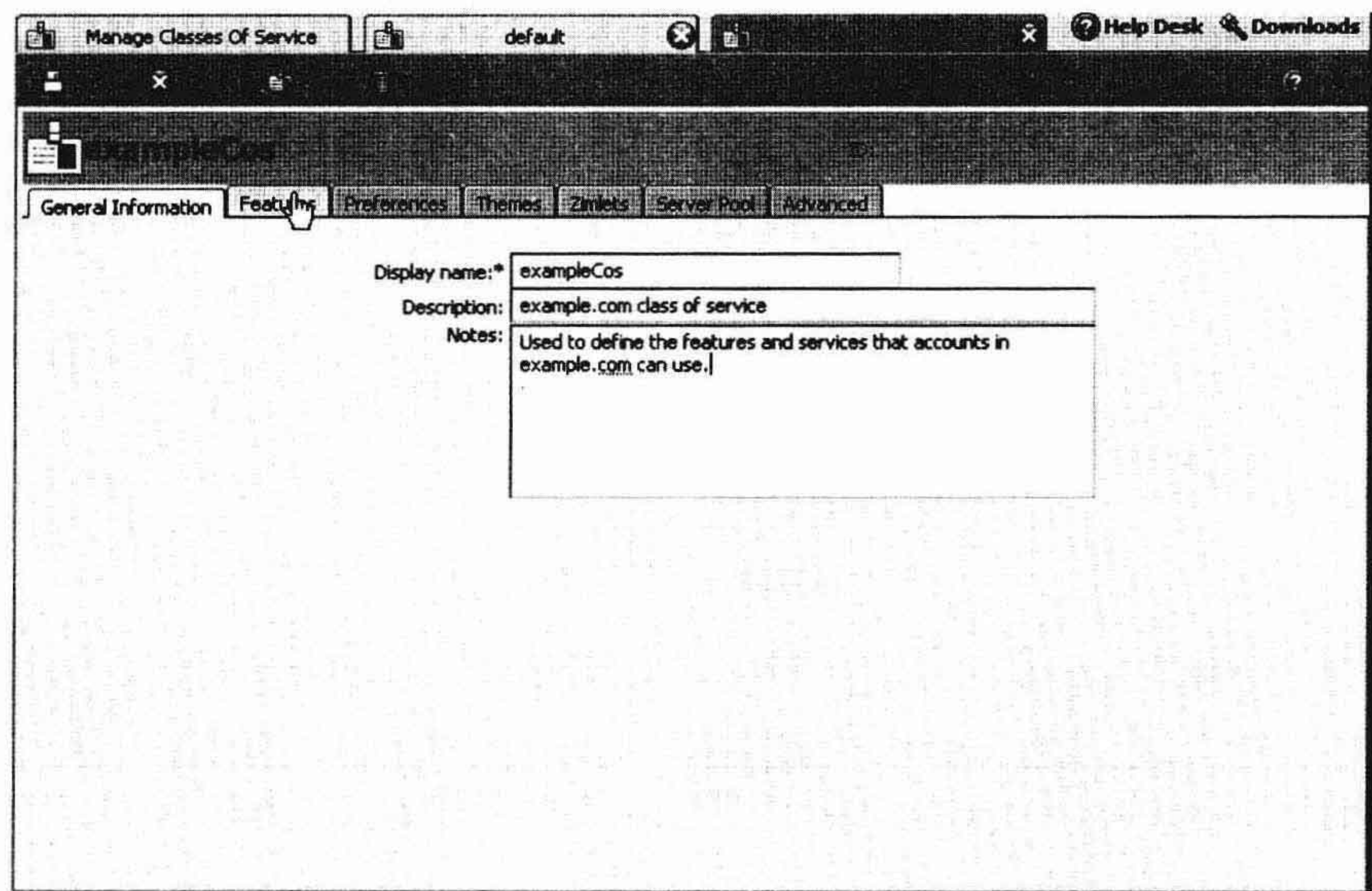


图 15-6 设置 CoS 的概要信息

在图 15-6 中，读者可以看到我们只是填写了一些概要信息，例如在 Display name 设置项填写 exampleCos 等。读者也可以查看其他几个标签页，它们将在本节的剩下部分被介绍到。

“General Information” 之后的下一个标签页是 “Features”，在该标签页上，选定域 example.com 中的用户可以使用哪些功能。

读者可以看到在图 15-7 中通过使用该 CoS，很容易选定供用户使用的各种功能。只要通过鼠标选择复选框中的选项来激活这些功能，用户就可以通过 Web 界面使用它们。这里选择所有功能，包括 “General Features” 标签页中的所有选项（虽然图中并没有显示出我们选择全部功能）。

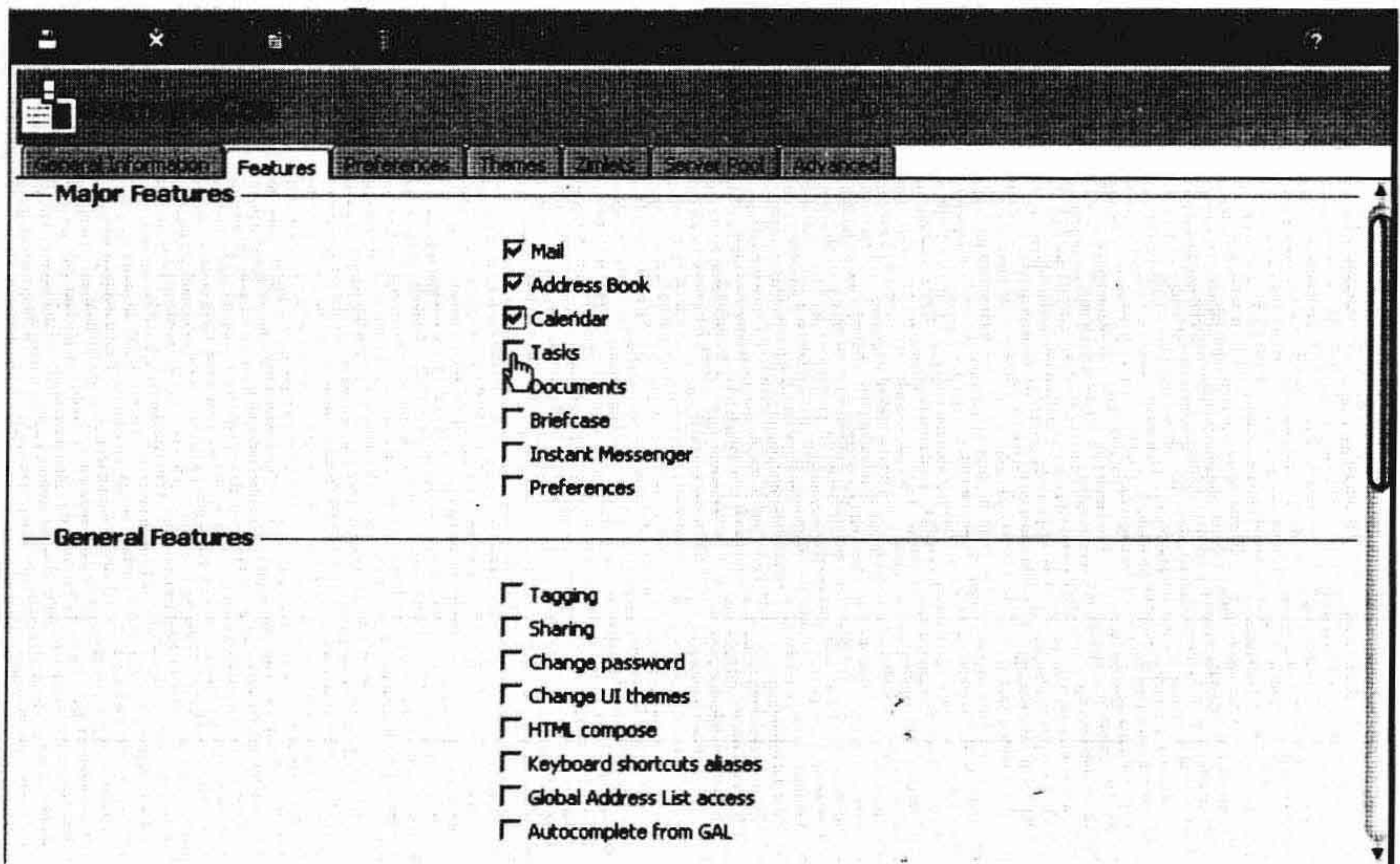


图 15-7 选择供用户使用的功能

下一步，添加一些用于 CoS 的首选项。在图 15-8 中，可以看到我们选定的选项子集。

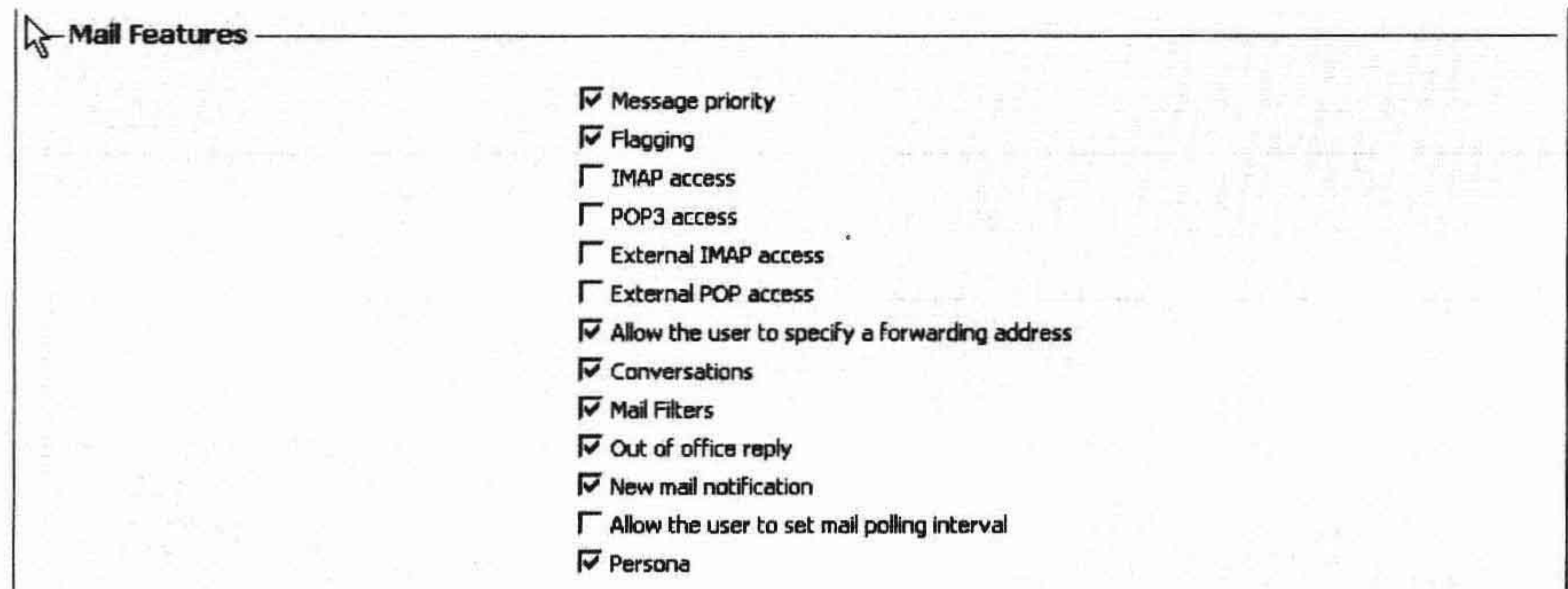


图 15-8 功能选择

图 15-8 里需要着重注意的一点是，不要允许外部的 IMAP 或者 POP 访问。这么做是因为希望公司里的每个用户使用 Web 界面收发邮件，而不要用各自的电子邮件客户端。因为只需为邮件服务开启 HTTPS 和 SMTP 端口，所以主机的管理变得更为简单，同时网络的复杂性也降低了。

在一些公司里，禁用电子邮件客户端是不可行的，因为有些员工需要用它进行离线作业。而离线作业是无法通过 Web 方式来完成的，所以在这种情况下，Zimbra 还提供了一个桌面客户端。可以从这里下载该客户端：<http://www.zimbra.com/products/desktop.html>。该客户端可通过 HTTPS 与 Zimbra 服务器通信，所以从网络的角度来看，它不需要额外的配置。

还是回到“Preferences”标签页，我们给出在 Web 浏览器中显示报文的方法。在图 15-9 中，设置“Mail Options”，其中一项是“Group mail by”。

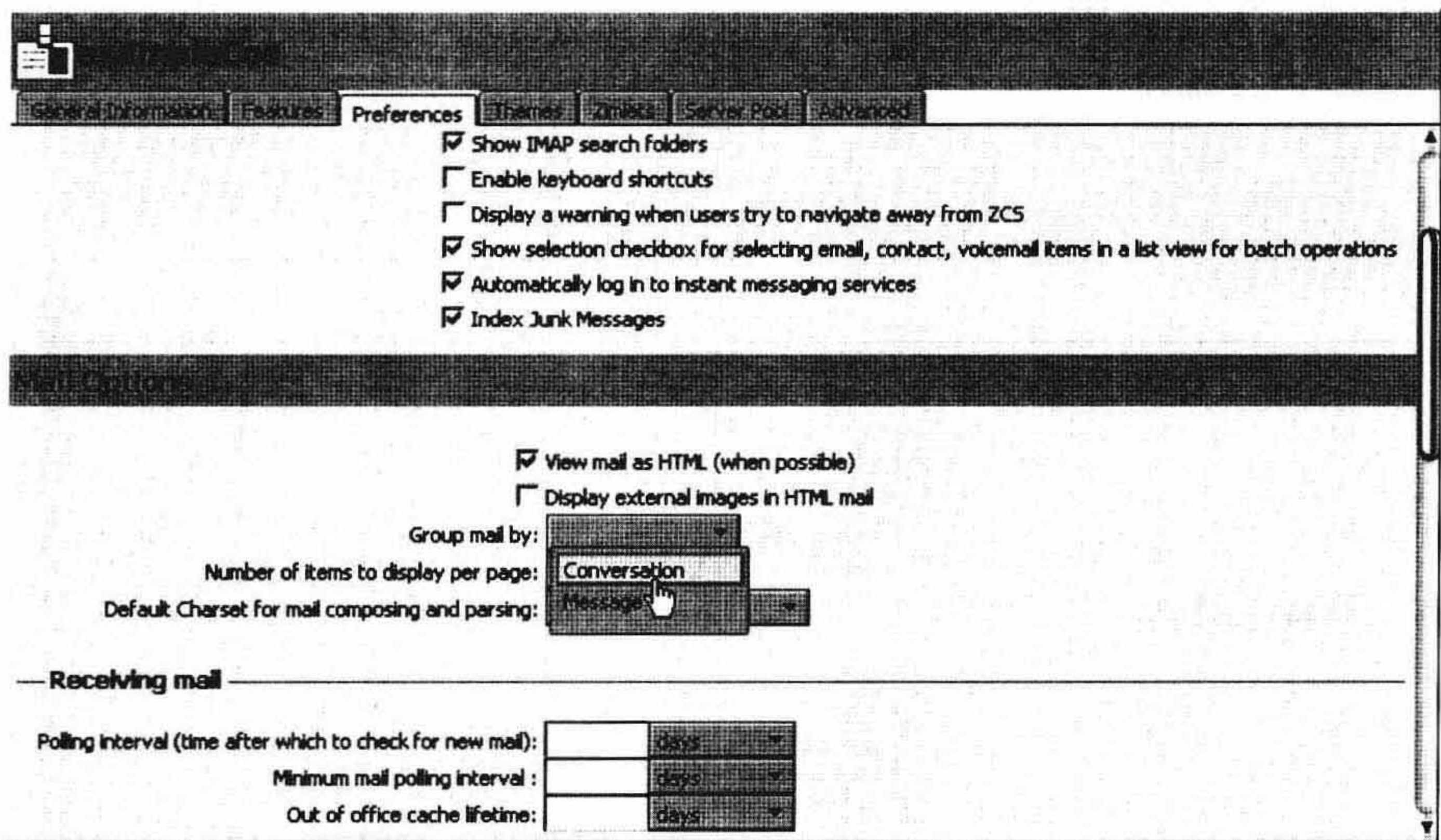


图 15-9 Preferences 标签页上的 Mail Options 部分

读者可能已经很熟悉根据会话对电子邮件进行分组的概念（也就是通常所说的讨论

模式), 举个例子, 就像 Google 的 Gmail 所做的那样。和传统的收件箱存放接收邮件、发件箱存放发送邮件不同, 邮件根据它们所属的会话被分组, 同一个会话下的邮件被分到同一组。所以, 如果我们发送一封邮件给某人, 然后收件人回复了, 这些邮件就会被分在同一组并在我们的邮件界面中一起被列出来。在稍后的“使用 Zimbra”小节里谈到 Web 界面的时候, 我们会详细展示这种邮件显示方式, 不过现在读者只需明白“Preferences”标签页是可以根据需要更改 Zimbra 默认行为的地方。“Preferences”标签页的其他设置项是默认字体、地址簿和日历。“Themes”标签页允许我们设置可用的用户主题并指定用户是否可以更改这些主题。读者也许想对此做限制, 以使得每个用户拥有一致的 Web 界面和使用体验, 不过这取决于读者的需要。图 15-10 展示了这里所做的设置。

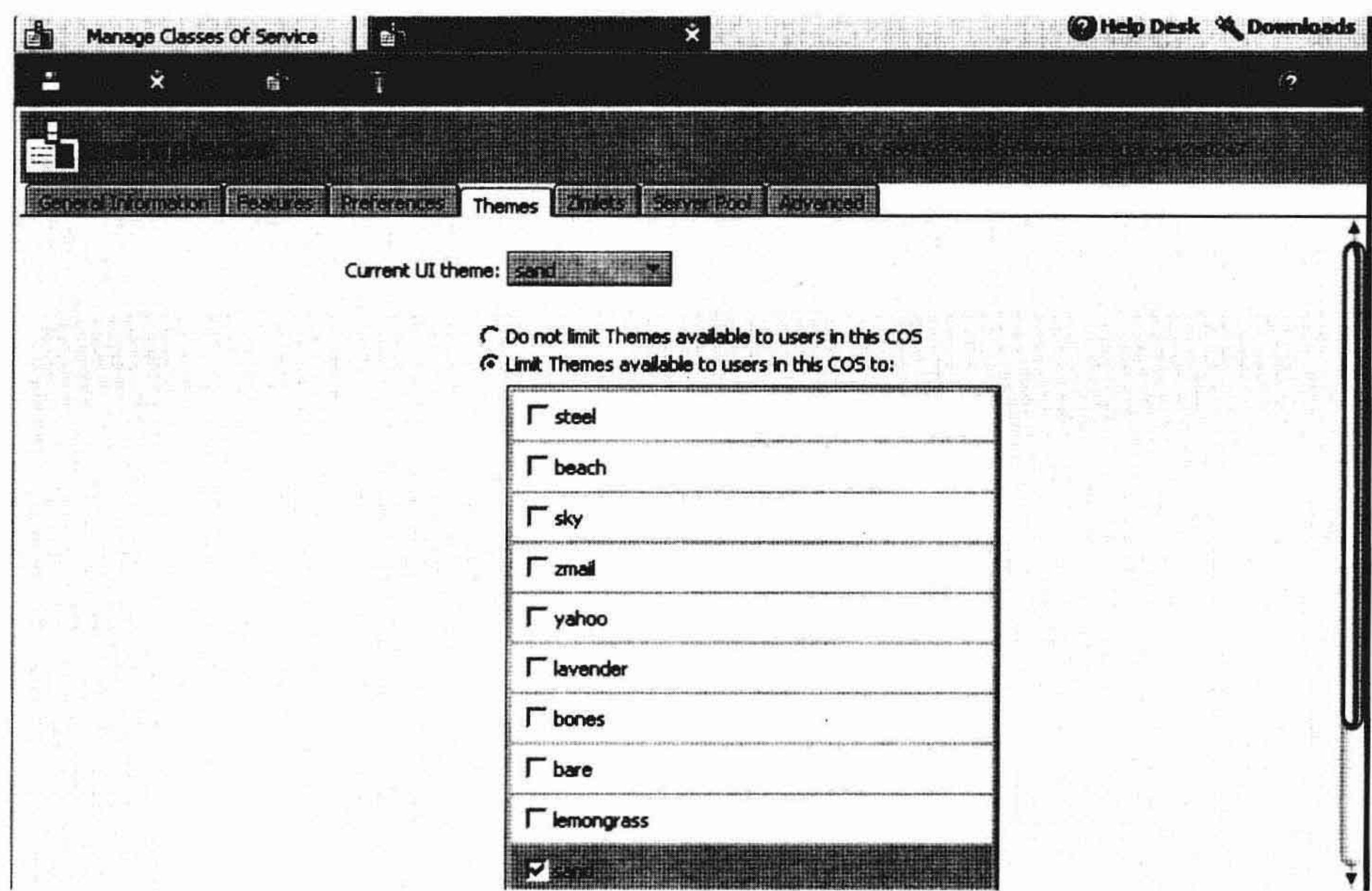


图 15-10 限制可用主题

在这里, 限制用户可用的主题只有一种: sand 主题, 这是默认主题。通过切换“Current UI theme”选项, 还可以设定当前使用的主题。

正如先前所解释过的, Zimlet 是由 Zimbra 公司或社区程序员开发的附加组件, 具有某种可用于 Web 界面的附加功能。可以在“Zimlets”标签页上设置可供用户使用 Zimlet, 如图 15-11 所示。

再次通过复选框选择 Zimlet, 使之可用。在图 15-11 中看到的 Zimlet 是随 Zimbra 一起默认安装的, 所以选择全部的 Zimlet。

下一个标签页是“Server Pool”。在该标签页上, 可以选择新建用户账号的归属服务器。在某些情况下, 可能在网络中安装多个邮件存储服务器, 并且希望特定用户利用一个特定的 CoS 使用一个特定的邮件服务器。在本例中只有一个邮件服务器供选择, 如图 15-12 所示。

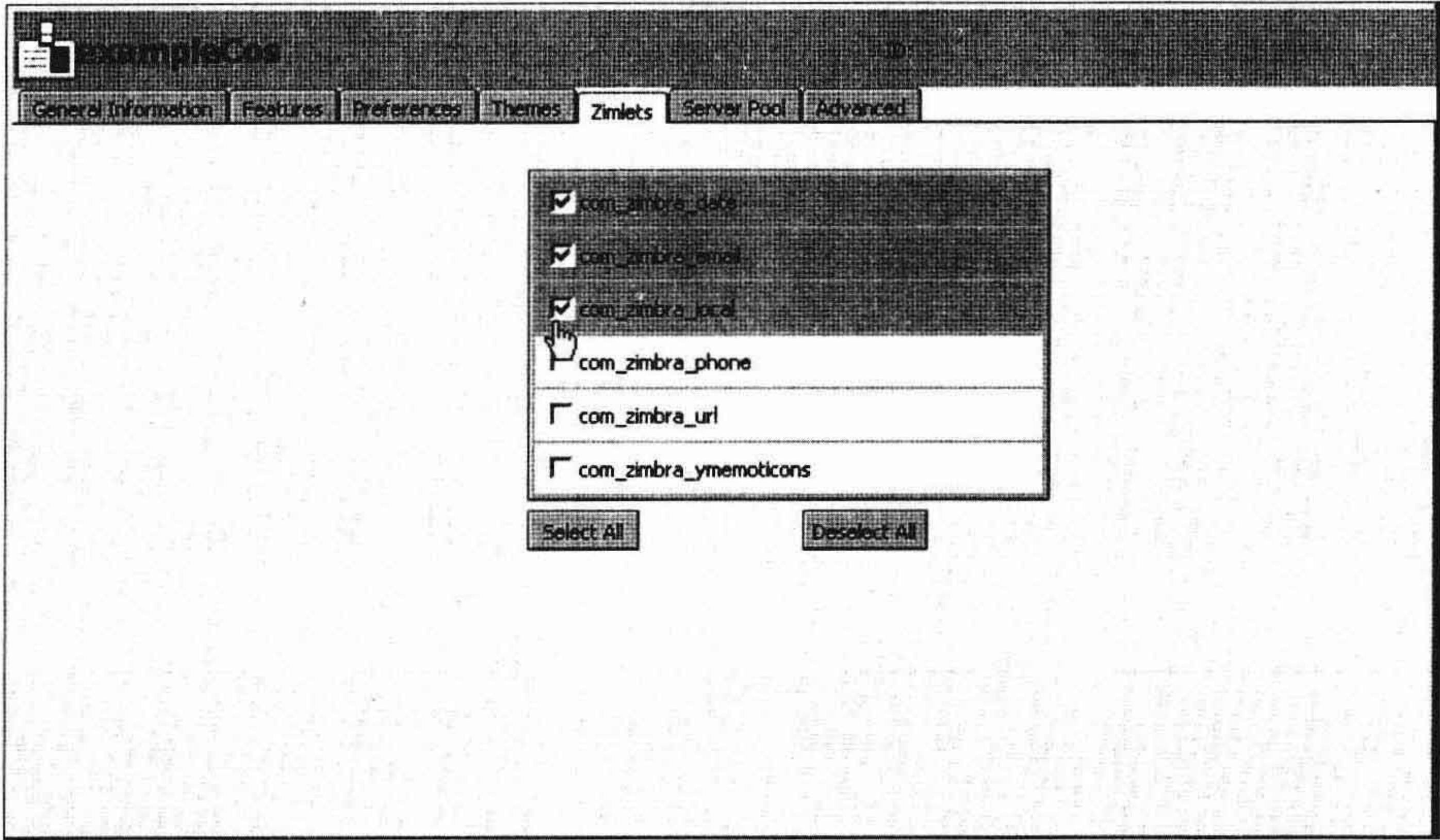


图 15-11 选择可用的 Zimlet

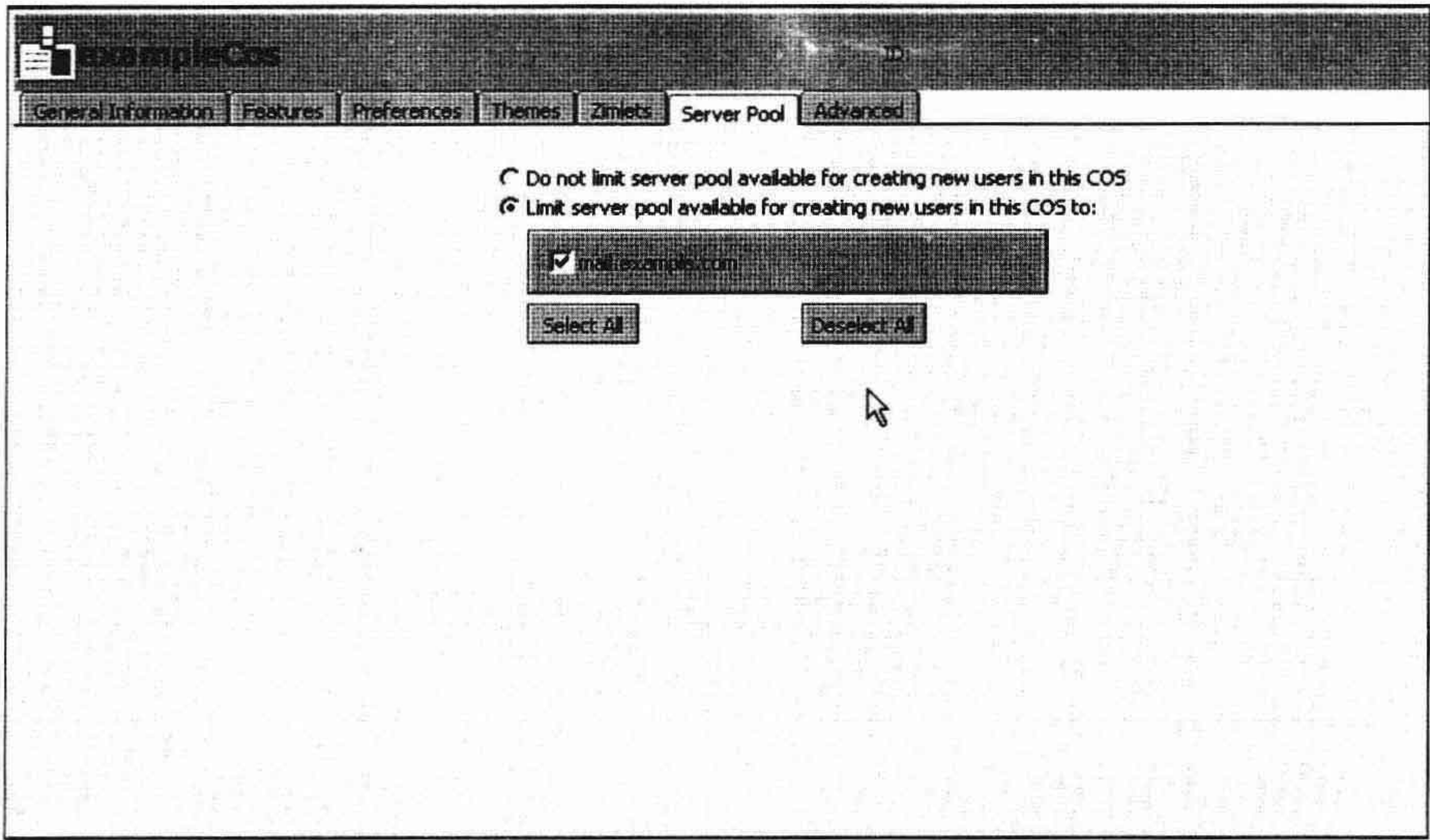


图 15-12 选择服务器组

下一个标签页是“Advanced”标签页。该标签页上的设置项包括邮箱空间上限、密码管理（长度、复杂度、有效期）和电子邮件保存策略。可以看到，图 15-13 把 500MB 的“Account quota”作为邮箱的空间上限，并设置了严格的密码管理策略。

这里对用户做了一些限制。除了 500MB 的邮箱空间上限外，用户在地址簿上只能记录 200 个联系人，当用户的邮箱空间利用率达到 80%，那么每隔两天会收到一封邮件通知。我们还设定密码的最短长度为 8 个字符，读者可以看到我们规定密码至少含有 2 个大写字母和 3 个小写字母。图 15-3 中还有其他一些类似的看不到的密码管理设置项，其中一项是密码的有效期，把它设为 90 天。“Advanced”标签页还可以设置其他的设置项，比如电子邮件的保存期限。使

用该设置项，我们可以统一设置垃圾邮件的保存期限，这样做可以减少主机上邮件占用的空间。

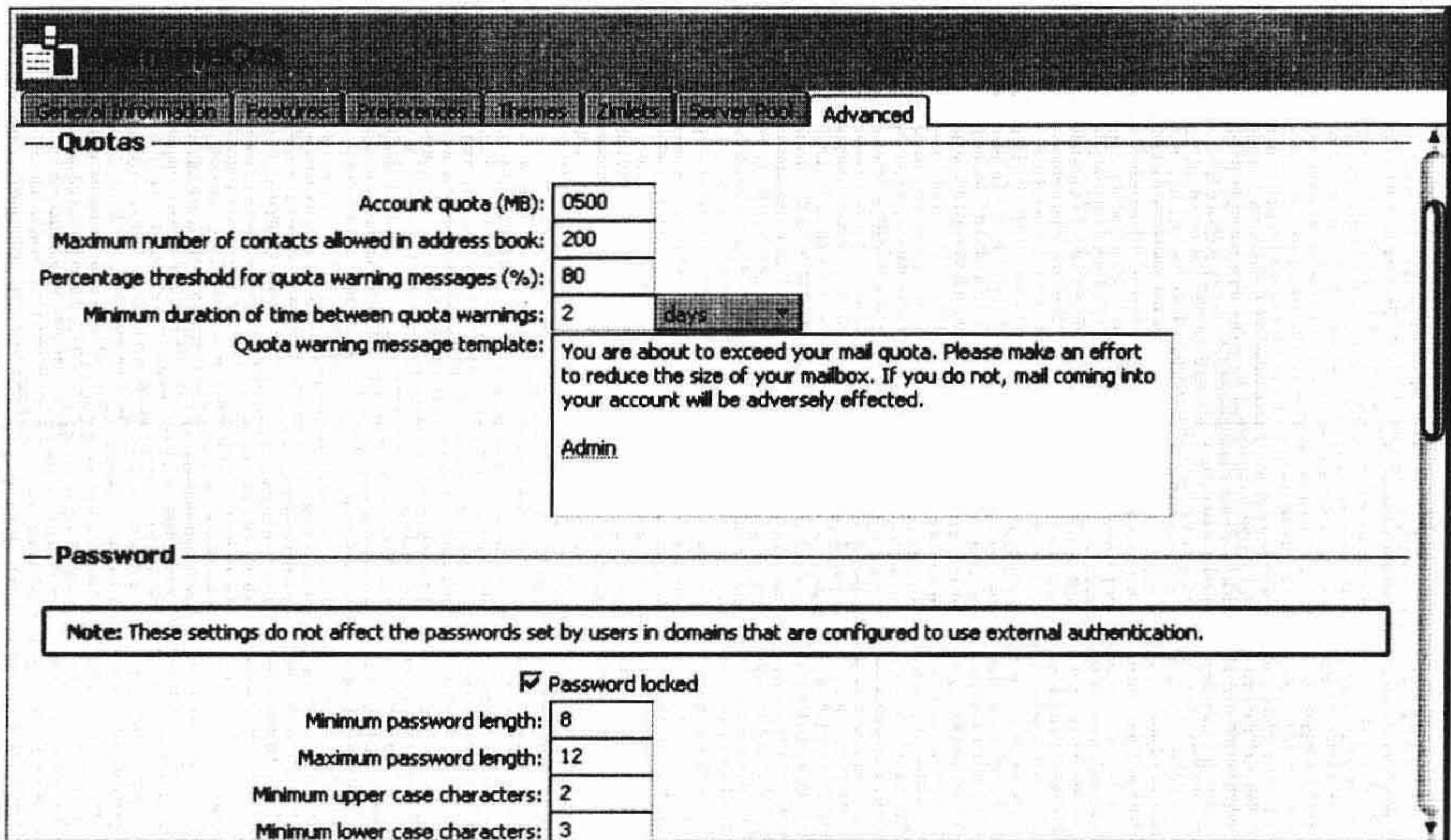


图 15-13 Advanced 标签页上的各项设置

现在定义好了一个 Class of Service，可以使用 Save 按钮保存该 CoS，并在创建新用户的时候用它或者把它赋予已经存在的用户。接下来创建一些用户。可以采用两种方式创建用户：一次创建一个用户，或者使用批量创建的方法一次性创建多个用户。

■注：zmprov 命令从技术上提供了第三种创建用户的方法。稍后我们会在本章的补充知识“zmprov：第三种方法”中谈到该命令。

15.2.8 添加新用户

现在讨论用户账号。首先单击左边菜单的“Accounts”选项。主界面立即显示当前的用户列表，如图 15-14 所示。

图 15-14 显示了 Zimbra 已经拥有 4 个用户：admin、ham.qiomilor、spam.syuqmdsha 和 wiki，这些用户是系统提供的账号。admin 就是当前登录的用户，该账号拥有管理 Zimbra 的高级权限。ham 和 spam 用户用于垃圾邮件过滤服务。Zimbra 使用这两个账号把收到的邮件分成垃圾邮件（令人讨厌的邮件）和正常邮件（有用的邮件）。正如读者所看到的，这两个账号的名字各有一串任意字符跟在一个点号（.）后面。Zimbra 这么做使得这两个账户的邮件地址不会轻易被垃圾邮件发送者猜出，从而用于降低过滤垃圾邮件的效率。Zimbra 使用 SpamAssassin 作为垃圾邮件过滤器。

■注：请仔细阅读第 10 章关于 SpamAssassin 如何工作的介绍。

最后一个用户是 wiki，他会被 Zimbra 服务器的 Documents 组件用到。稍后会在“Using Zimbra”小节里谈到这方面的内容。

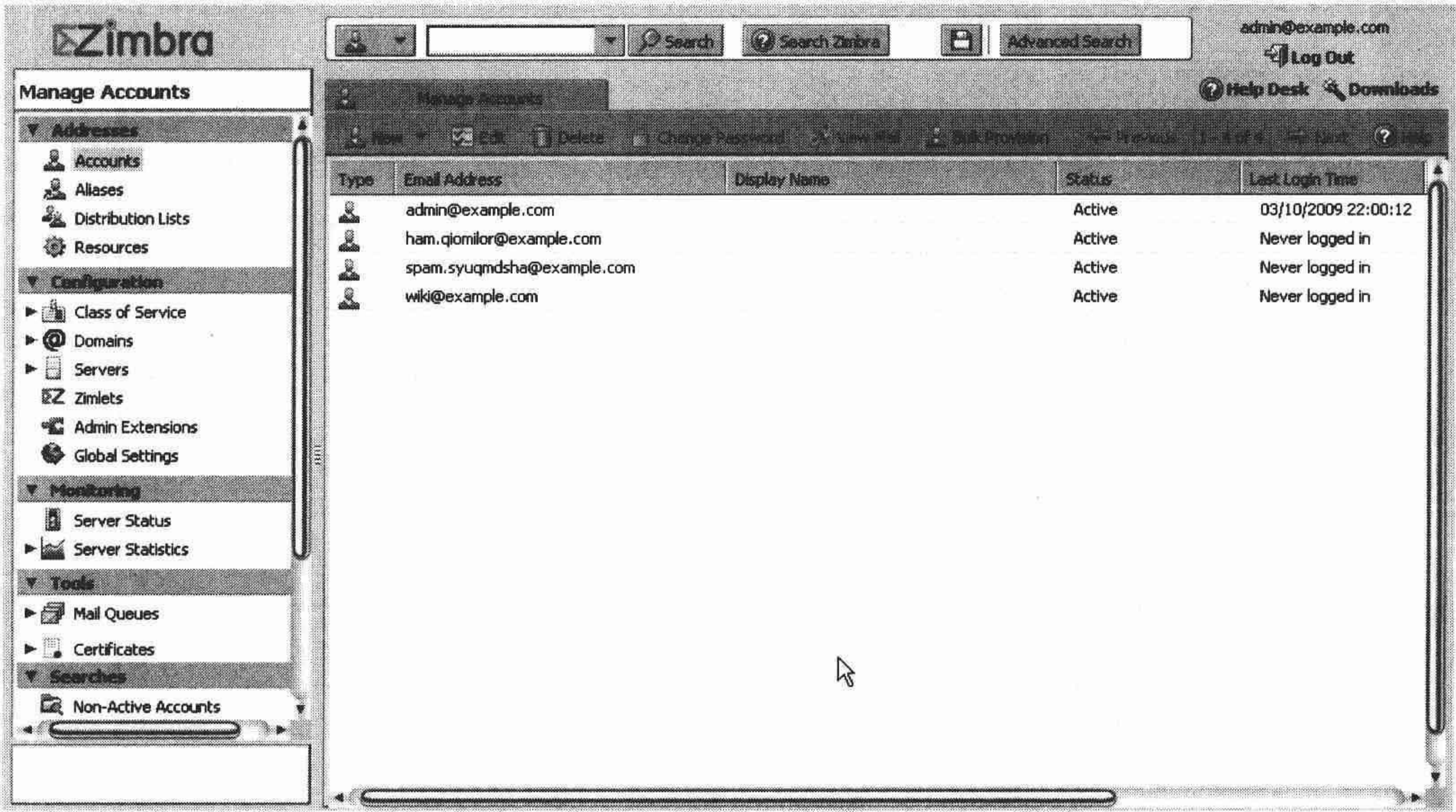


图 15-14 选中控制台上的 Accounts 选项

现在添加一些用户。如前所述，Zimbra 允许每次添加一个用户或者用批量创建方法一次性添加多个用户。在管理控制台下，可以用 Bulk Provision（批量创建）按钮添加多个用户账号。单击该按钮会弹出一个对话框要求输入一个 CSV（CSV 是 comma-separated values 的简写，即逗号分隔值）文件的路径名。Zimbra 的批量创建功能允许为每个用户指定一个邮箱地址、一个显示名称和一个密码。该 CSV 文件如下所示。

```
jsmith@example.com,Jane Smith,somepassword
bsingh@example.com,Bev Singh,somepassword
ataylor@example.com,Angela Taylor,somepassword
```

把这些信息保存在一个叫做 users.txt 的文件中。在图 15-15 中，可以看到我们已经找到该文件的绝对路径并准备单击“Next”按钮。

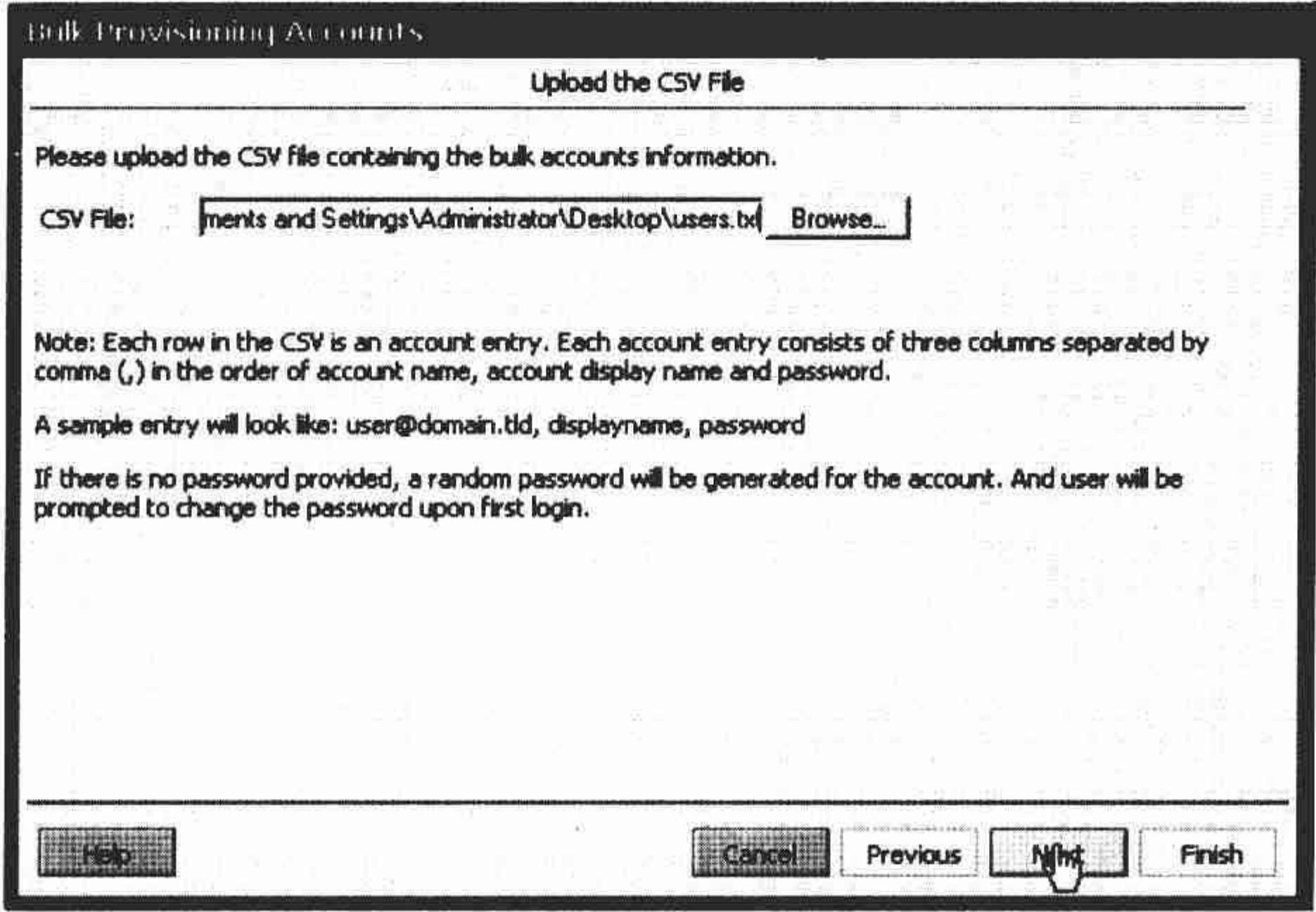


图 15-15 添加 users.txt 文件用于批量创建

单击“Next”按钮，就会看到所要录入的账号信息。下面准备添加该 CSV 文件列出的各个用户，如图 15-16 所示，并可以通过该对话框确认用户信息是否正确。

当单击“Next”按钮时，创建操作即刻开始。我们会看到一个确认窗口，该窗口列出所添加的用户账号和它们的创建操作的完成状态。到这一步，所有用户的创建操作全部成功完成，如图 15-17 所示。

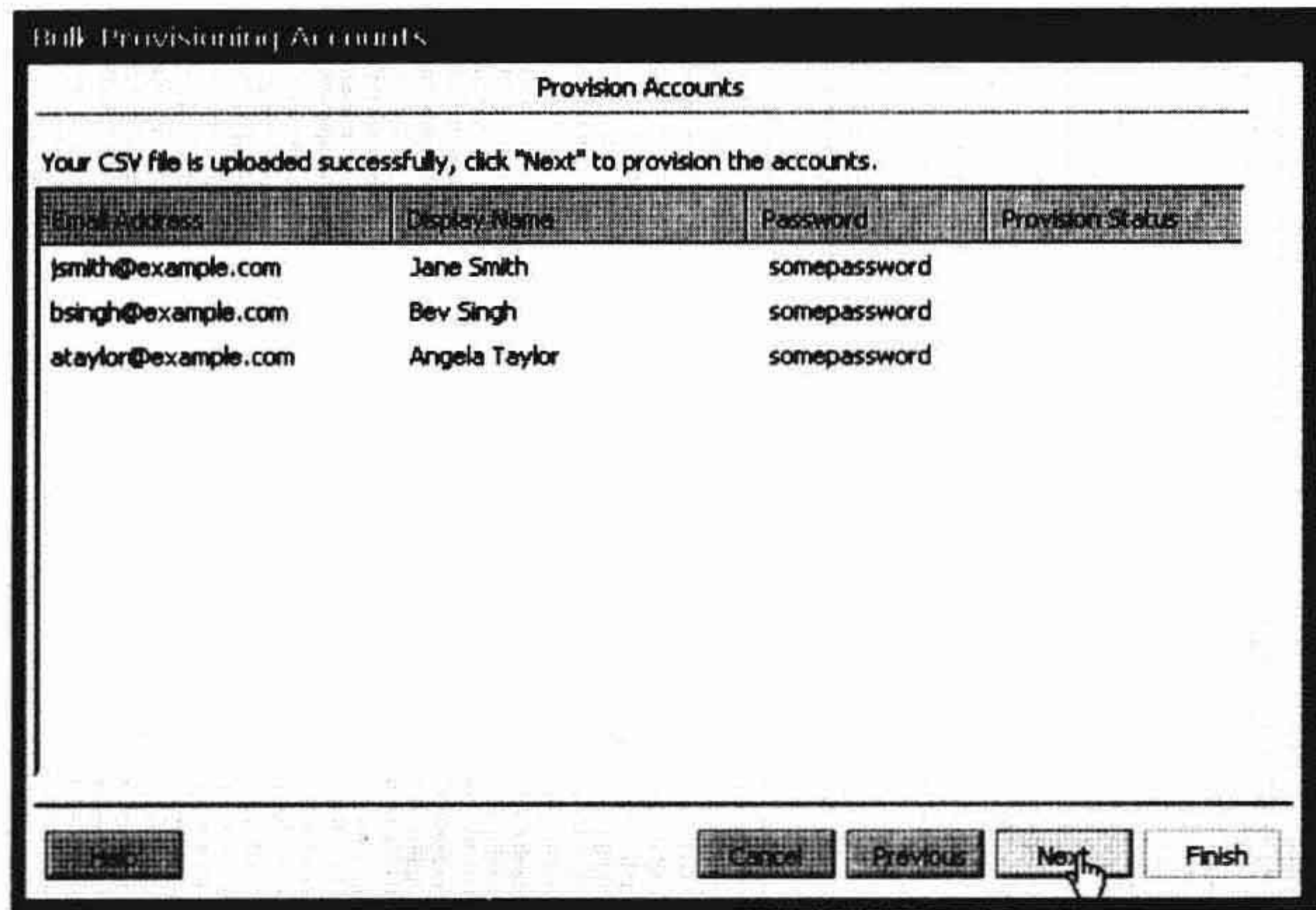


图 15-16 在添加用户之前确认其信息

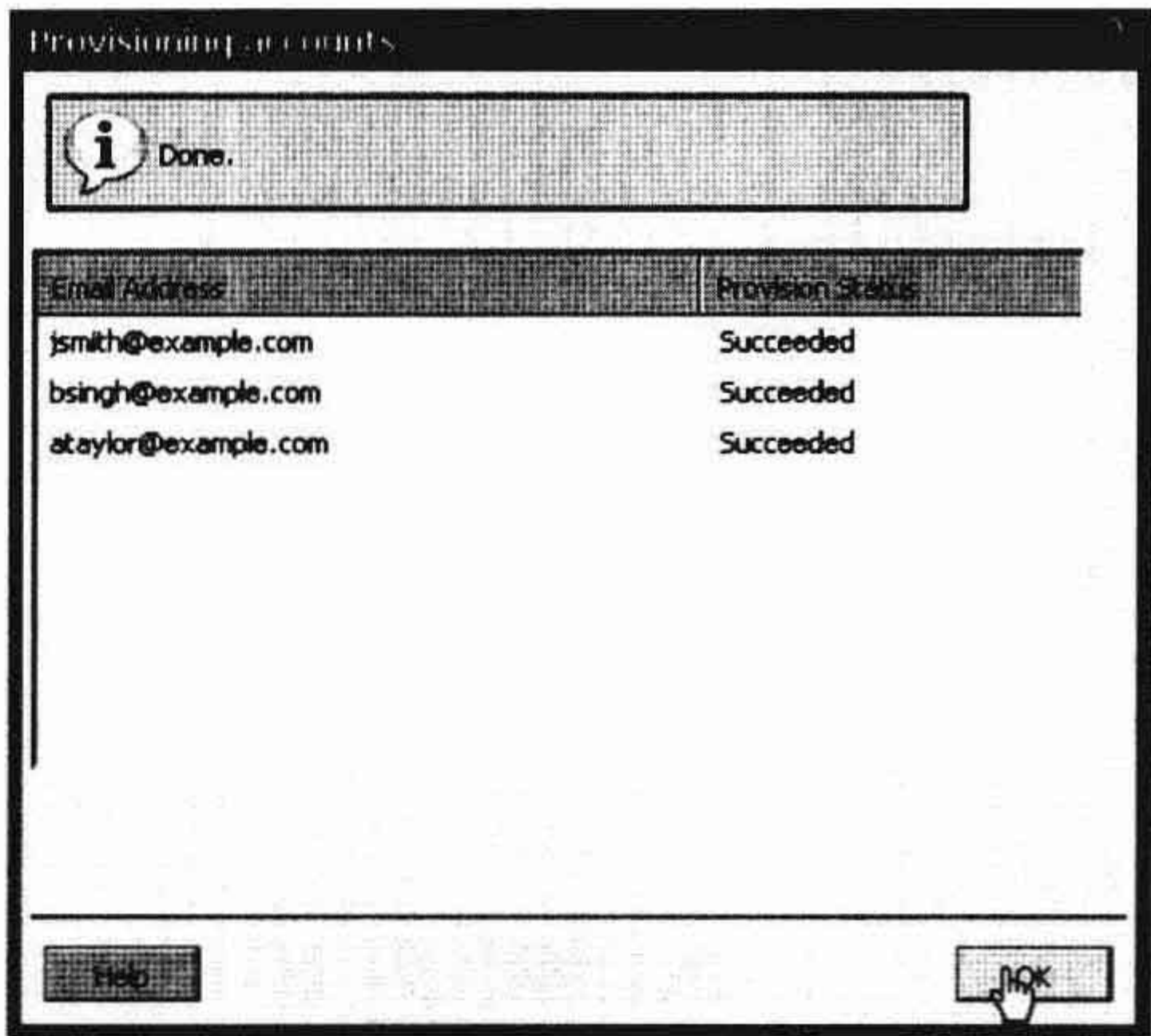


图 15-17 当前成功创建的用户

现在结束创建用户过程，我们看到最后一个弹出窗口，该窗口显示创建步骤的汇总信息，如图 15-18 所示。

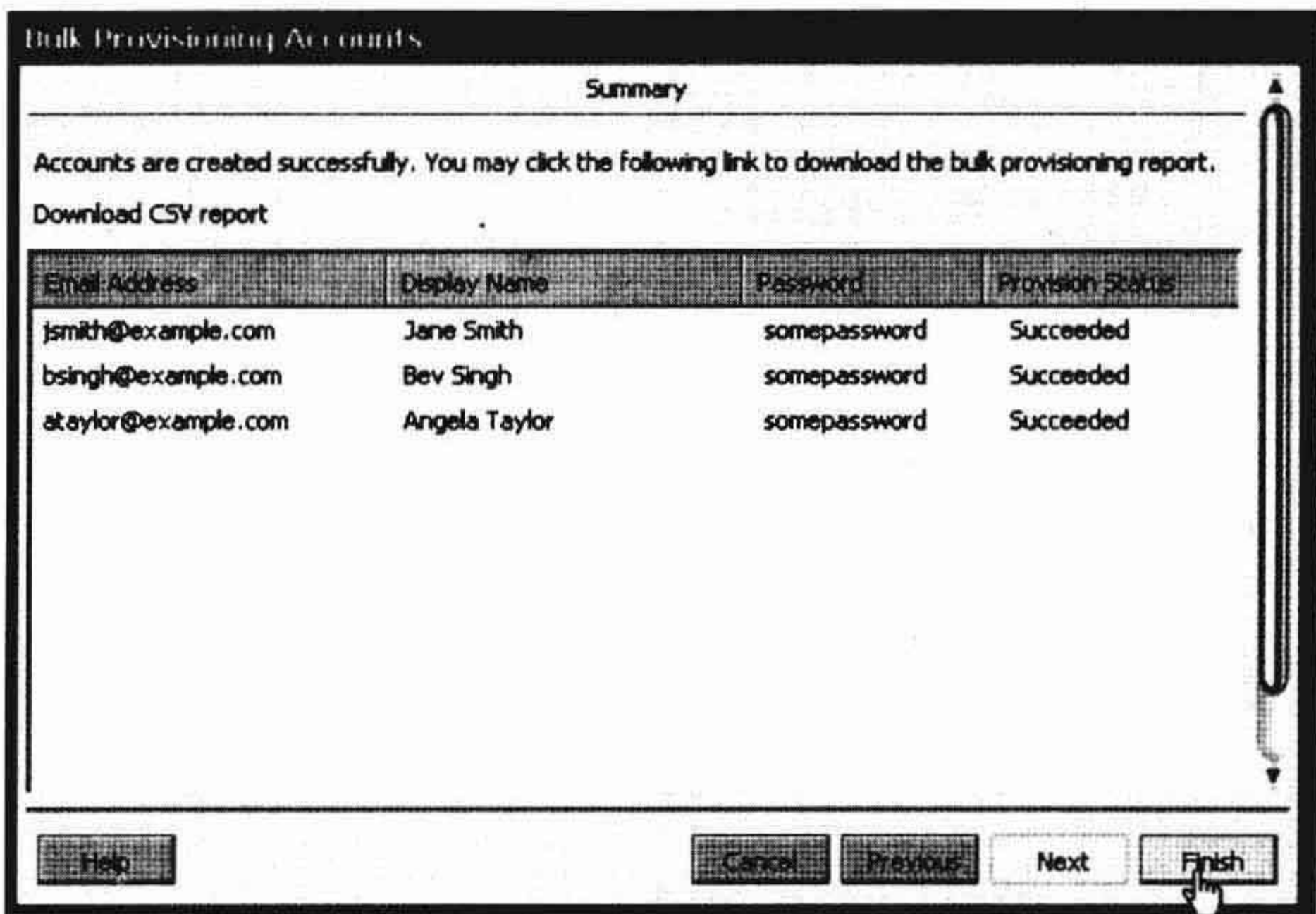


图 15-18 批量创建过程的汇总信息

接下来向读者介绍如何添加单个用户。首先单击“Accounts”窗口中的“New”按钮，如图 15-19 所示。

接下来准备创建用户 Jim Bob。为这个新用户添加详细信息，如账号名（用于登录 Zimbra 服务器），还有用户的名字和姓，如图 15-20 所示。当用户发送邮件时，Canonical address（规范地址）会出现在表单字段里。读者还会看到我们指定该用户的 CoS 为 examplecos。Class of Service 字段会自动寻找任何可用的 CoS。

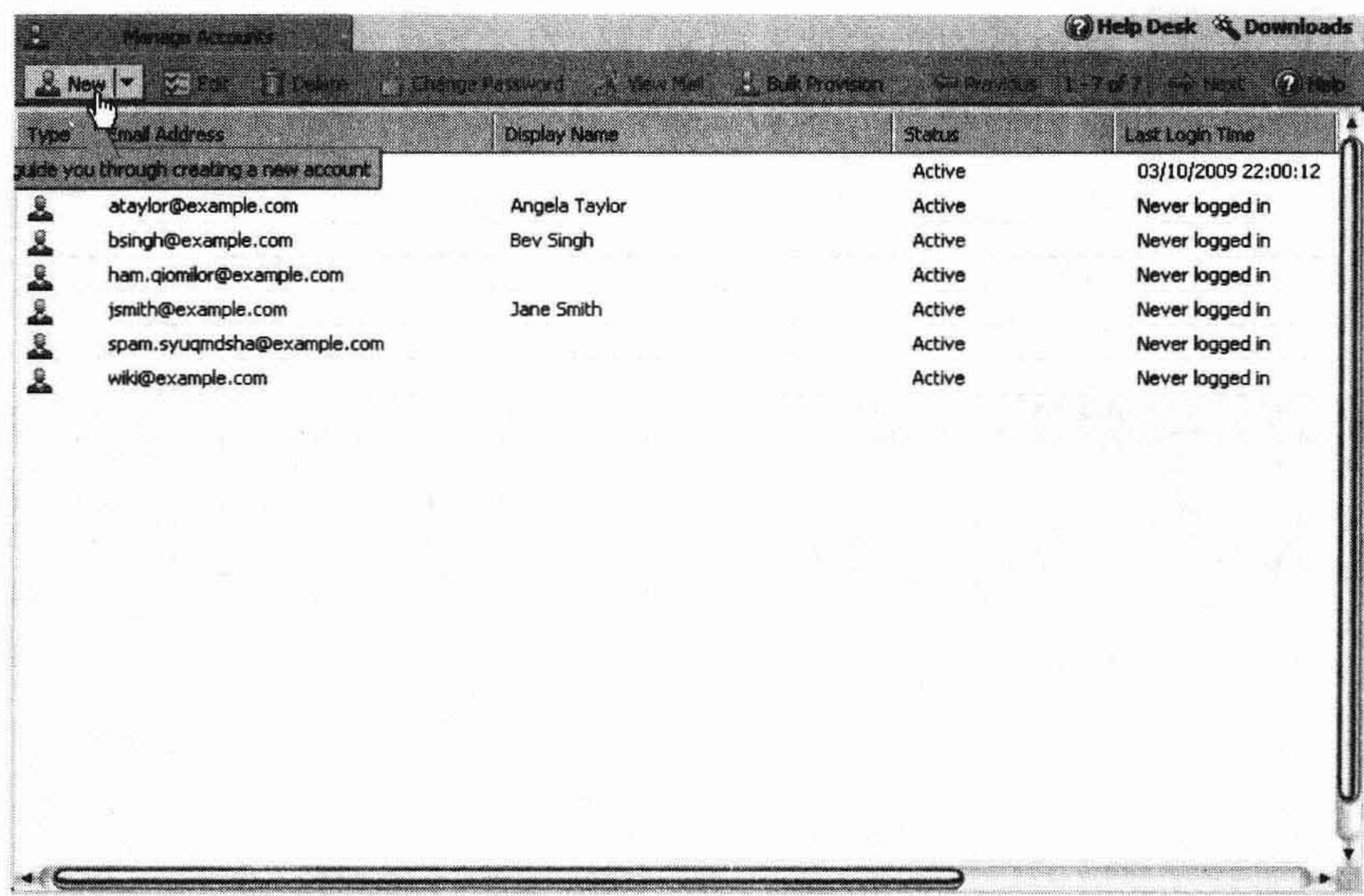


图 15-19 添加单个用户

The screenshot shows the 'New Account' form with the 'General Information' tab selected. The form is divided into two main sections: 'Account Name' and 'Account Setup'. In the 'Account Name' section, there are fields for 'Account name' (filled with 'jbob'), 'First name' (filled with 'Jim'), 'Middle initial' (empty), 'Last name' (filled with 'Bob'), 'Display name' (filled with 'Jim Bob'), 'Canonical address' (filled with 'jim.bob@example.com'), and a 'Hide in GAL' checkbox. In the 'Account Setup' section, there are fields for 'Account status' (set to 'Active'), 'Class of Service' (filled with 'examplecos'), and a dropdown menu (filled with 'examplecos'). There are 'Cancel', 'Previous', and 'Next' buttons at the bottom.

图 15-20 添加用户的账号信息

下一步是添加用户的联系人信息。可以在图 15-21 中看到我们输入的联系人信息。

The screenshot shows the 'New Account' form with the 'Contact Information' tab selected. The form contains fields for 'Phone' (filled with '+61 333 999 333'), 'Company' (filled with 'Example Pty Ltd'), 'Department' (filled with 'Sanitation'), 'Office' (filled with 'Melbourne'), 'Address' (filled with '44 Spring St Melbourne'), and 'City' (filled with 'Melbourne'). There are 'Cancel', 'Previous', 'Next', and 'Finish' buttons at the bottom.

图 15-21 普通的联系人信息

接下来需要为新用户添加邮箱别名。给用户 Jim Bob 添加邮箱别名 jim.bob@example.com 和 jimmy.bob@example.com。邮箱别名用于把多个邮箱地址绑定到一个用户账号上。现在，用户 Jim Bob 可以接收邮件的地址为 jim.bob@example.com、jimmy.bob@example.com 和 jbob@example.com。可以在图 15-22 中看到邮箱别名的设置。

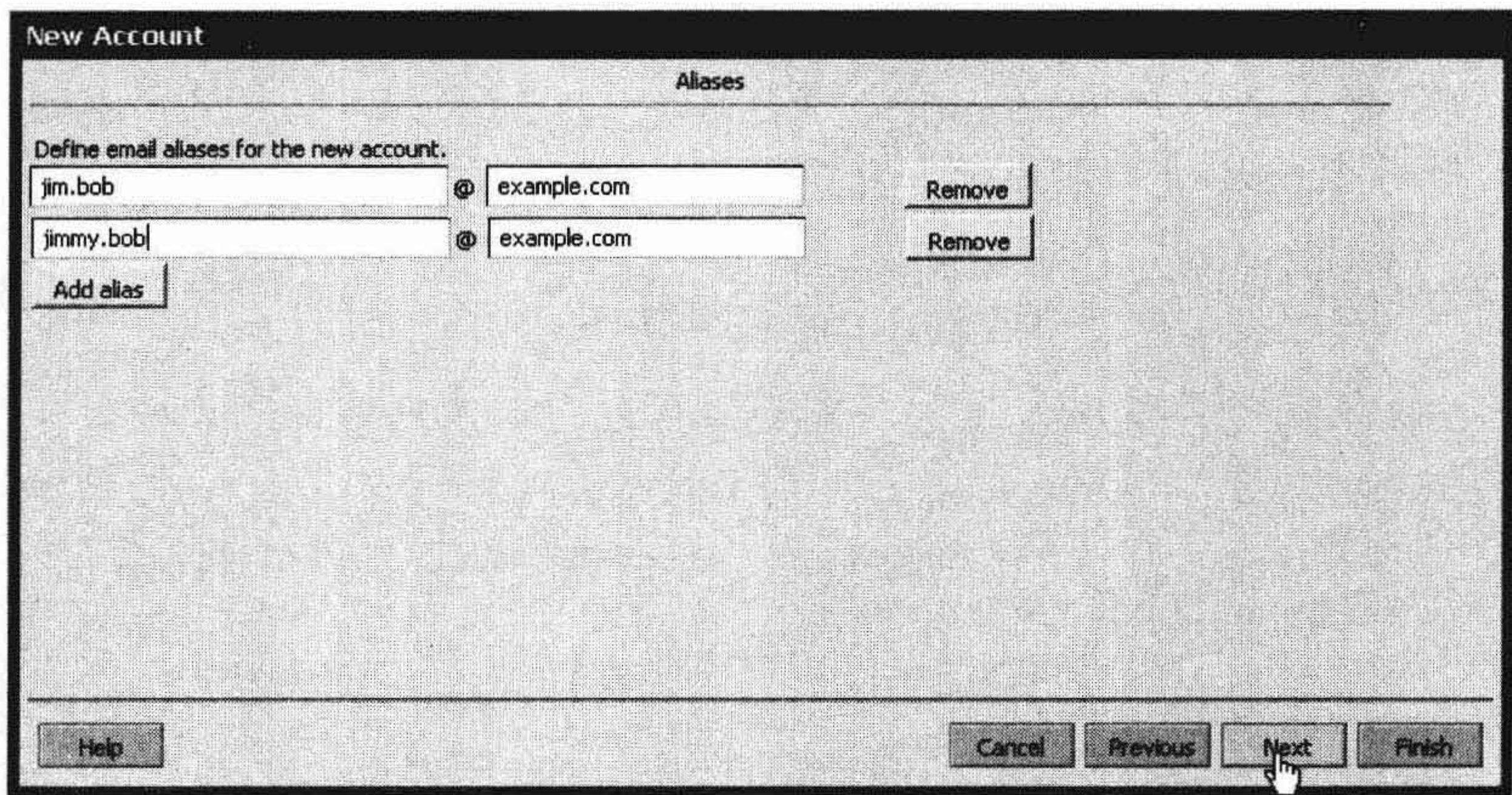


图 15-22 用户邮箱别名

单击“Next”按钮进入下一个窗口，在此可以设置邮件转发地址。如果复选框中的“Allow the user to specify a forwarding address”项被勾选，则当一个用户登录其账号的时候，他可以设置自己的一个邮件转发地址。该设置允许在需要的情况下，把一封发给该新建用户的邮件转发到另一个邮箱地址。这里还可以设置把邮件隐式地转发到另一个邮箱地址而不会引起原收件用户的注意。这种方法可用于查看发往用户的邮件。我们不需要上述任何一种设置，所以不选择这些选项，如图 15-23 所示。

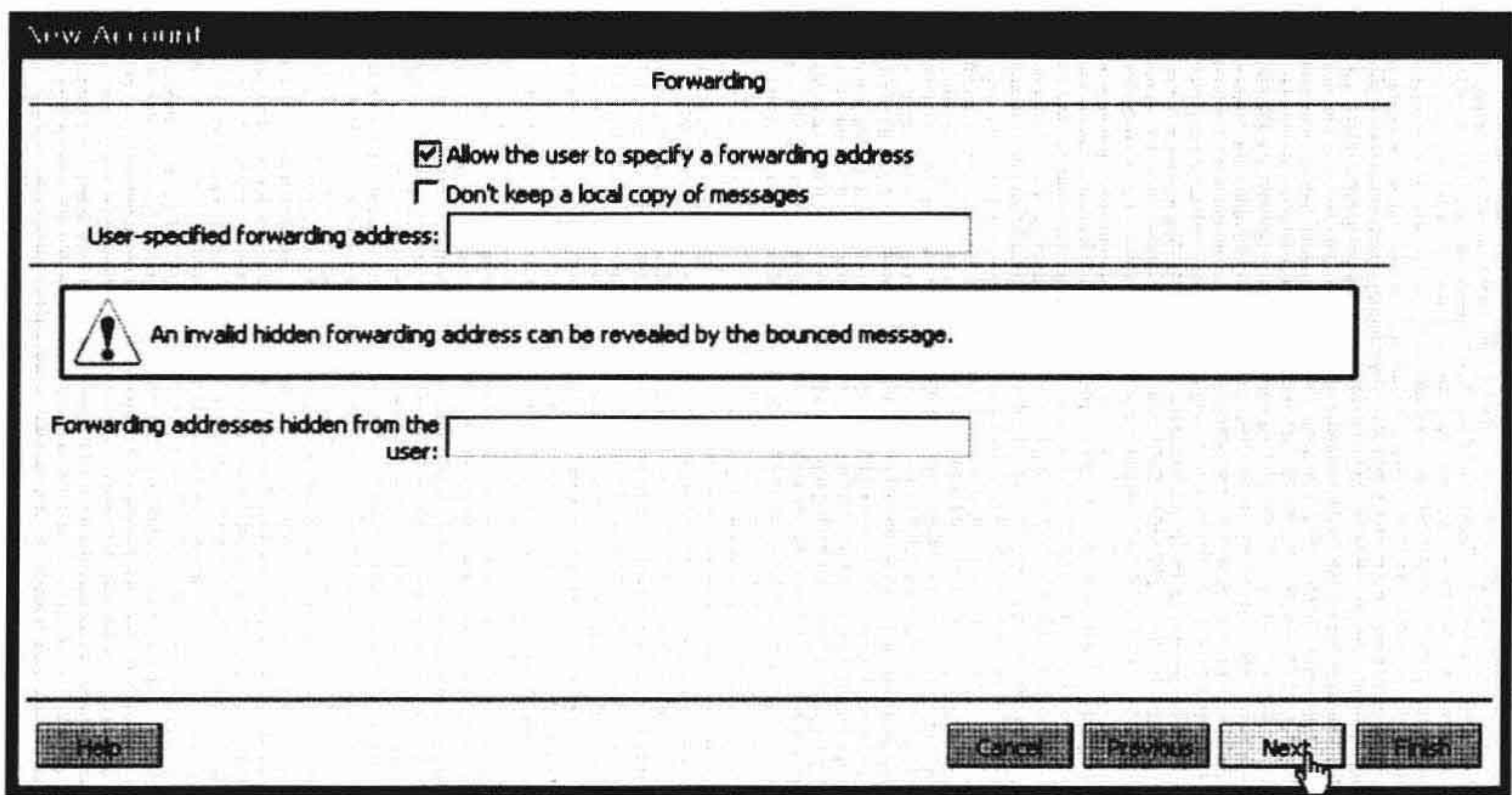


图 15-23 邮件转发地址

现在单击“Finish”按钮，剩下的配置信息由 examplecos Cos 自动填入。新用户 Jim Bob 已成功创建，并和其他已创建用户账号一起出现在 Accounts 窗口中。

ZMPROV：第三种方法

在本章前面已经提到过，从技术上讲有第三种方法可以批量创建用户账号。可以用 `zmprov` 命令一次性添加大量用户，包括进行 CoS 设置和其他需要的特定设置。

```
zmprov ca jsmith@example.com password displayName 'Jane Smith' zimbraCOS 5e9069c4-4e3b-466e-b64c-03ca442e0247
```

这里使用附带 `ca` 参数的 `zmprov` 命令创建用户账号。设置新用户 `jsmith@example.com` 的密码为 `password`，显示名为“Jane Smith”。通过指定 `zimbraCoS`，把该用户添加到 `examplecos` CoS。在管理控制台上或者使用下列命令可以从 `examplecos` 的内容中找到 `zimbraCoS` 的信息。

```
$ zmprov gc examplecos |grep zimbraId
zimbraId: 5e9069c4-4e3b-466e-b64c-03ca442e0247
```

那么为了一次性创建大量用户账号，需要执行包含以下内容的脚本。

```
#!/bin/bash
zmprov ca bsingh@example.com password displayName 'Bevan Singh' zimbraCOSId 5e9069c4-4e3b-466e-b64c-03ca442e0247
zmprov ca jsmith@example.com password displayName 'Jane Smith' zimbraCOSId 5e9069c4-4e3b-466e-b64c-03ca442e0247
zmprov ca ataylor@example.com password displayName 'Angela Taylor' zimbraCOSId 5e9069c4-4e3b-466e-b64c-03ca442e0247
```

如果把上述代码行添加到一个脚本文件中，例如 `users.sh`，然后以 `zimbra` 用户的身份执行该脚本，就可以把这 3 个用户添加到 `examplecos` CoS。要获得关于使用 `zmprov` 命令的更多内容，请阅读以下网页中的内容，见“http://www.zimbra.com/docs/os/latest/administration_guide/A_app-command-line.13.2.html”。

15.2.9 邮箱别名和邮件分发列表

“Accounts”菜单下面是“Aliases”菜单。单击“Aliases”菜单，可以看到之前为用户 Jim Bob 创建的两个邮箱别名。在这里，可以添加新的邮件别名或者修改原有的邮件别名。读者可以在图 15-24 中看到我们创建的两个新的邮件别名。

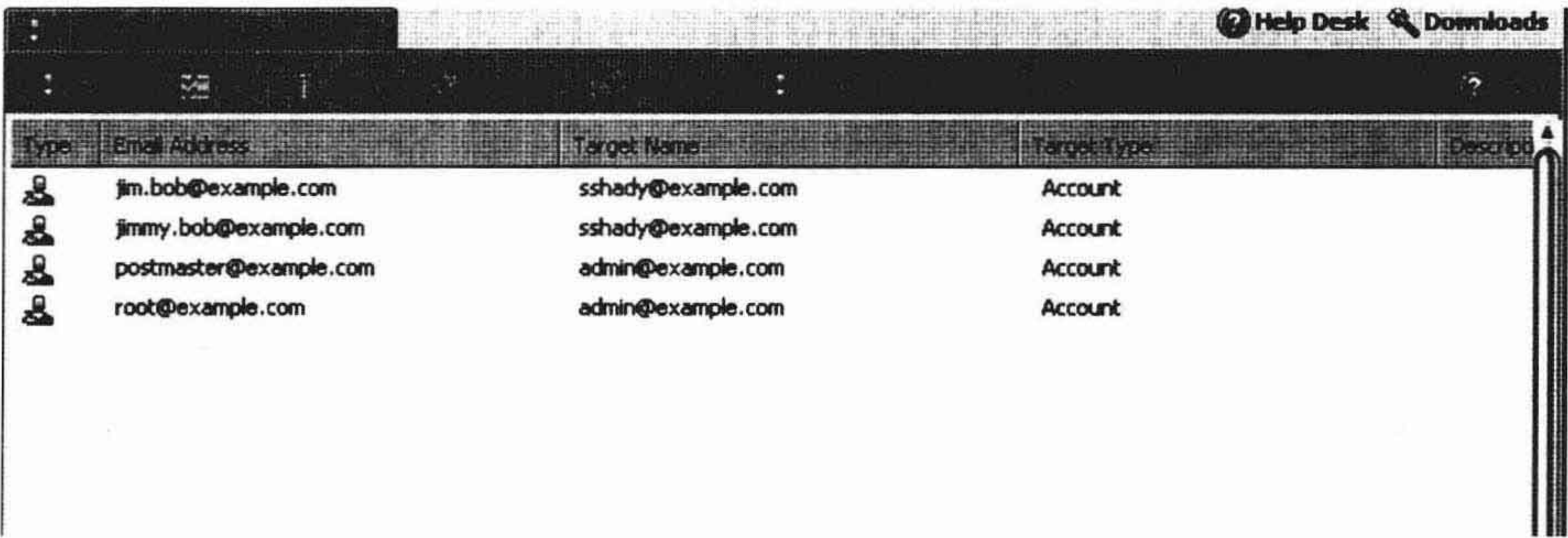


图 15-24 我们创建的新的邮件别名

当前 `root` 和 `postmaster` 的邮箱地址都是 `admin` 用户的邮箱别名。`root` 和 `postmaster` 邮箱地址是用于主机和 Internet 之间通信的重要邮箱地址。主机使用 `root@example.com` 邮箱地址发送警告信息，而 `postmaster@example.com` 邮箱地址则被主机用于和众多的 Internet 服务进

行通信。也可以把这两个邮箱地址所属的用户账号更改为另一个更适合处理这些邮件的用户账号，如果不这么做，就必须记得经常登录 admin 账号阅读这些邮件。

现在准备添加两个邮件分发列表。邮件分发列表就是把多个邮箱地址组成一组当成一个地址使用，非常类似于用一个邮箱别名代替多个邮箱地址。下面准备创建一个叫做 allstaff 的邮件分发列表和一个叫做 melboffice 的邮件分发列表。allstaff 邮件列表用来给所有职员发送邮件，而 melboffice 邮件列表只用来给工作在 Melbourne 公司的职员发送邮件。可以创建任意想要的邮件分发列表，但每个列表的名称都必须是唯一的。我们可以为所有的销售职员、业务经理和主管等人员创建一个名叫 sales 的邮件分发列表，这使得我们可以联系公司里的多个业务部门。要创建一个邮件分发列表，首先单击“Distribution List”，然后单击“New”按钮，如图 15-25 所示。

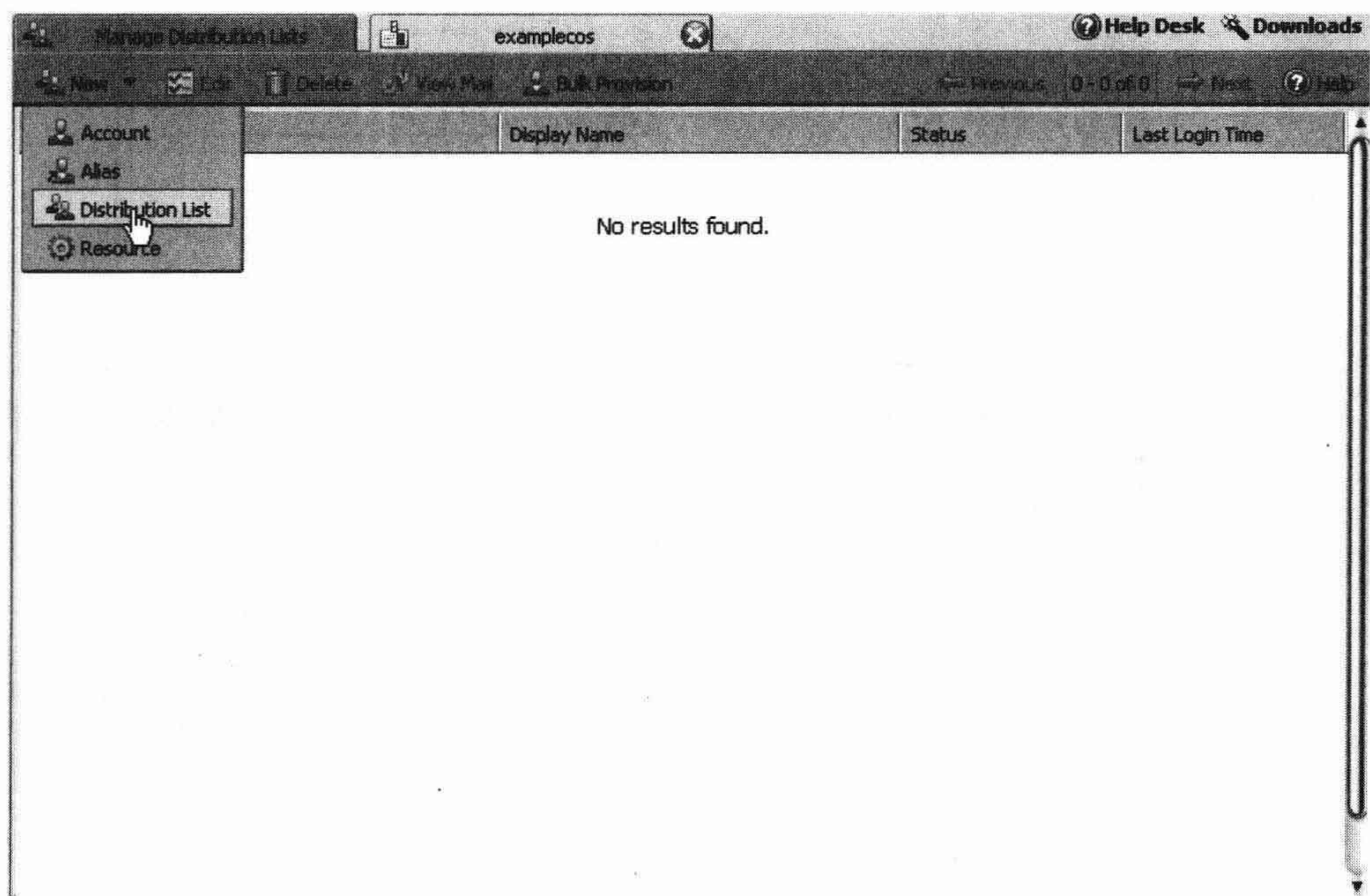


图 15-25 创建一个新的邮件分发列表

创建一个邮件分发列表，并赋予它唯一的邮箱名（该邮箱名不能是邮箱域中已有的邮箱名）。我们准备创建一个叫做 allstaff@example.com 的邮件分发列表，在图 15-26 中，可以看到我们为创建该列表所填写的信息。

可以采用以下两种方法之一把用户成员添加到列表：在“Or enter addresses below”输入框中手动输入它们的邮箱地址，或者使用搜索工具。如果不输入任何搜索字段就单击“Search”按钮，则返回所有可添加到列表的用户信息。在图 15-26 中可以看到，我们已经选中一些需要添加到该列表的用户。在此不打算添加系统上的任何用户，如 ham、spam 和 wiki 等账号。当高亮选中将要添加到邮件分发列表的用户时，单击“Add”按钮，该操作将把选中的用户移到“List Members”显示框中，如图 15-27 所示。

在更为复杂的应用场景里，可以把邮件分发列表添加到其他邮件分发列表，还可以给列表取别名。单击“Save”按钮，保存该邮件分发列表，然后再次单击“Distribution List”菜

单查看保存好的邮件分发列表；图 15-28 显示了系统中的邮件分发列表。

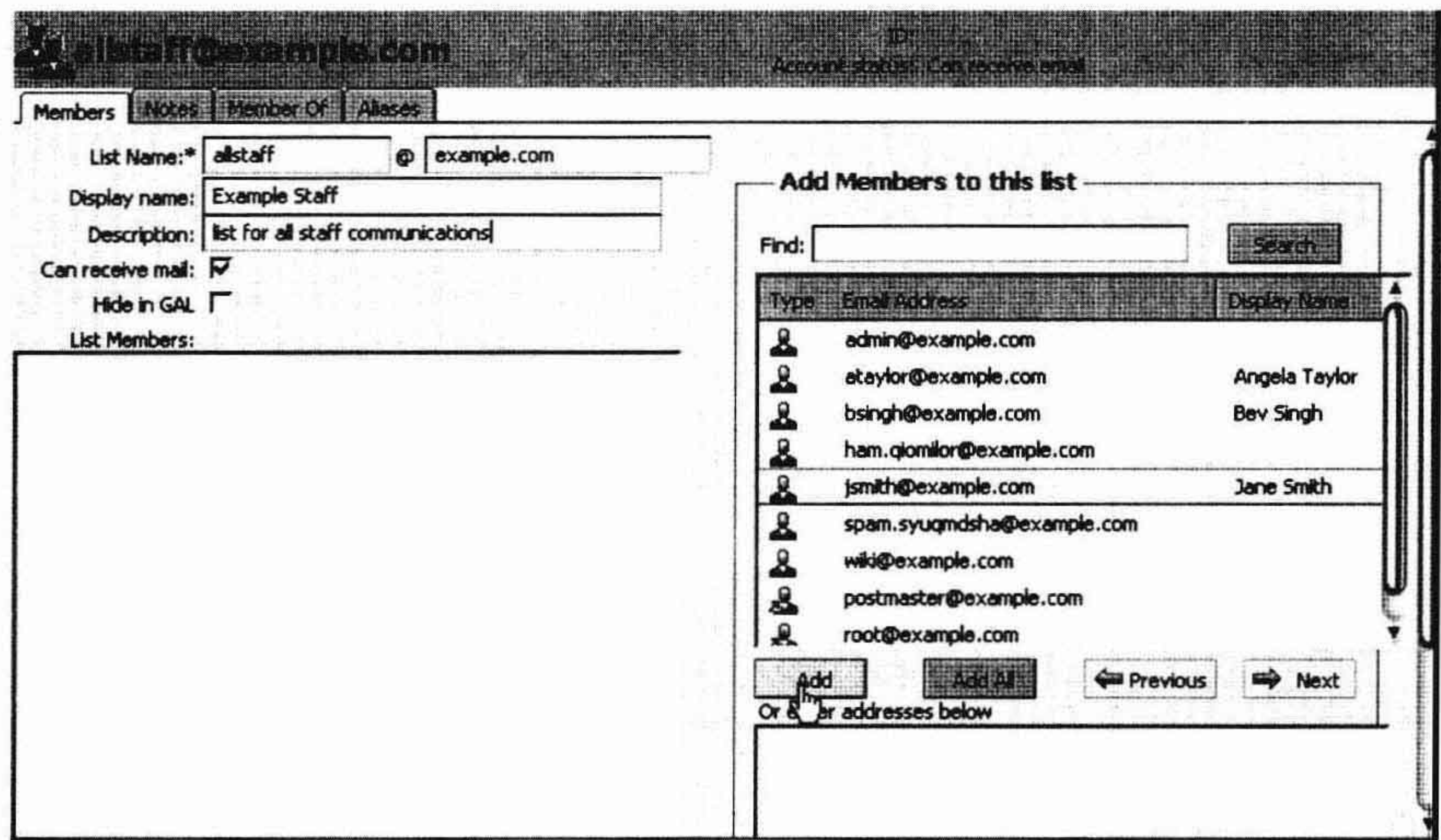


图 15-26 添加用户邮箱到邮件分发列表

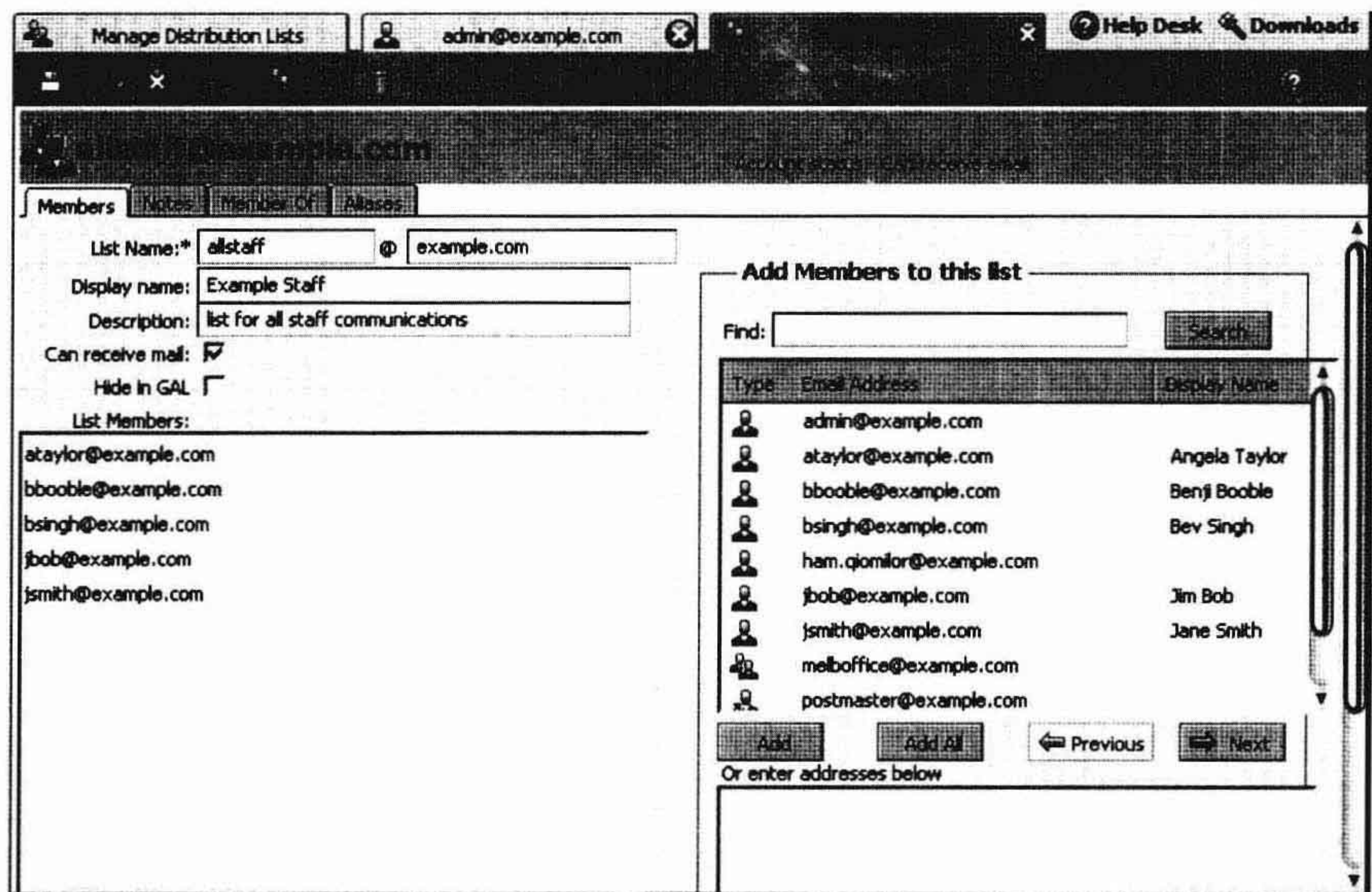


图 15-27 邮件分发成员列表

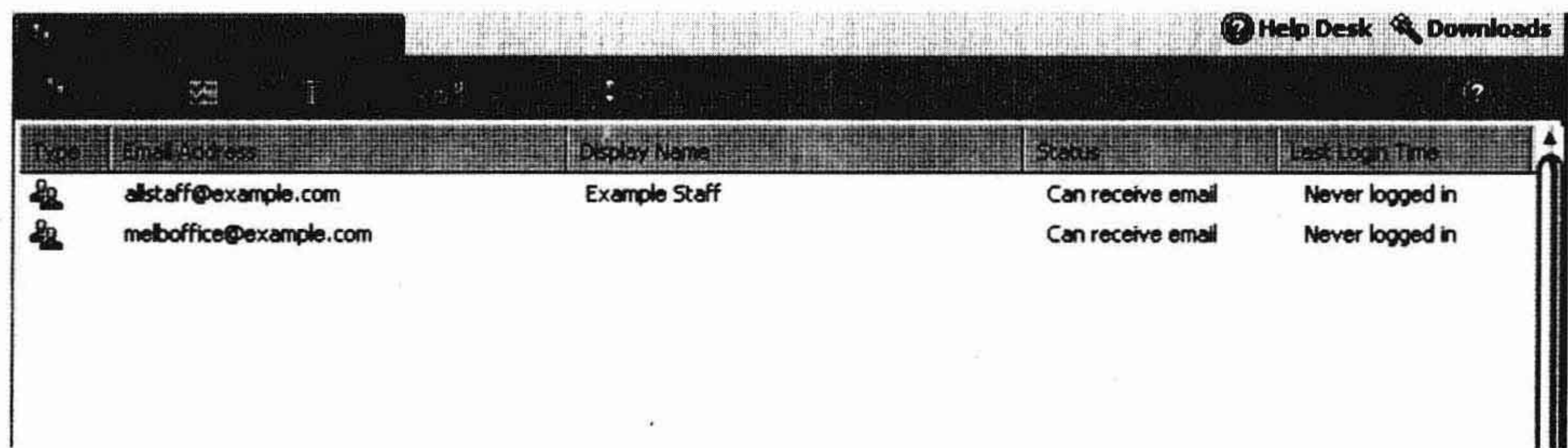


图 15-28 我们的邮件分发列表

15.2.10 添加资源

我们不仅可以创建用户账号、邮箱别名和邮件分发列表，还可以创建资源。资源就是想要跟踪其使用状况的各种东西，如房间、投影机以及公司里其他需要登记才能使用的公用设备。如果公司有会议，则可以在日程表（稍后在“使用 Zimbra”一节将介绍）中通知职员参加会议。当安排好一个会议，就可以预约一些资源，如房间或者投影机，这样别人就可以知道什么人将在什么时候使用什么资源。

下面准备向读者介绍如何创建一个房间资源。选中“Resources”菜单，弹出当前所有可用资源列表。然后单击“New”按钮，弹出“New Resource”窗口，如图 15-29 所示。

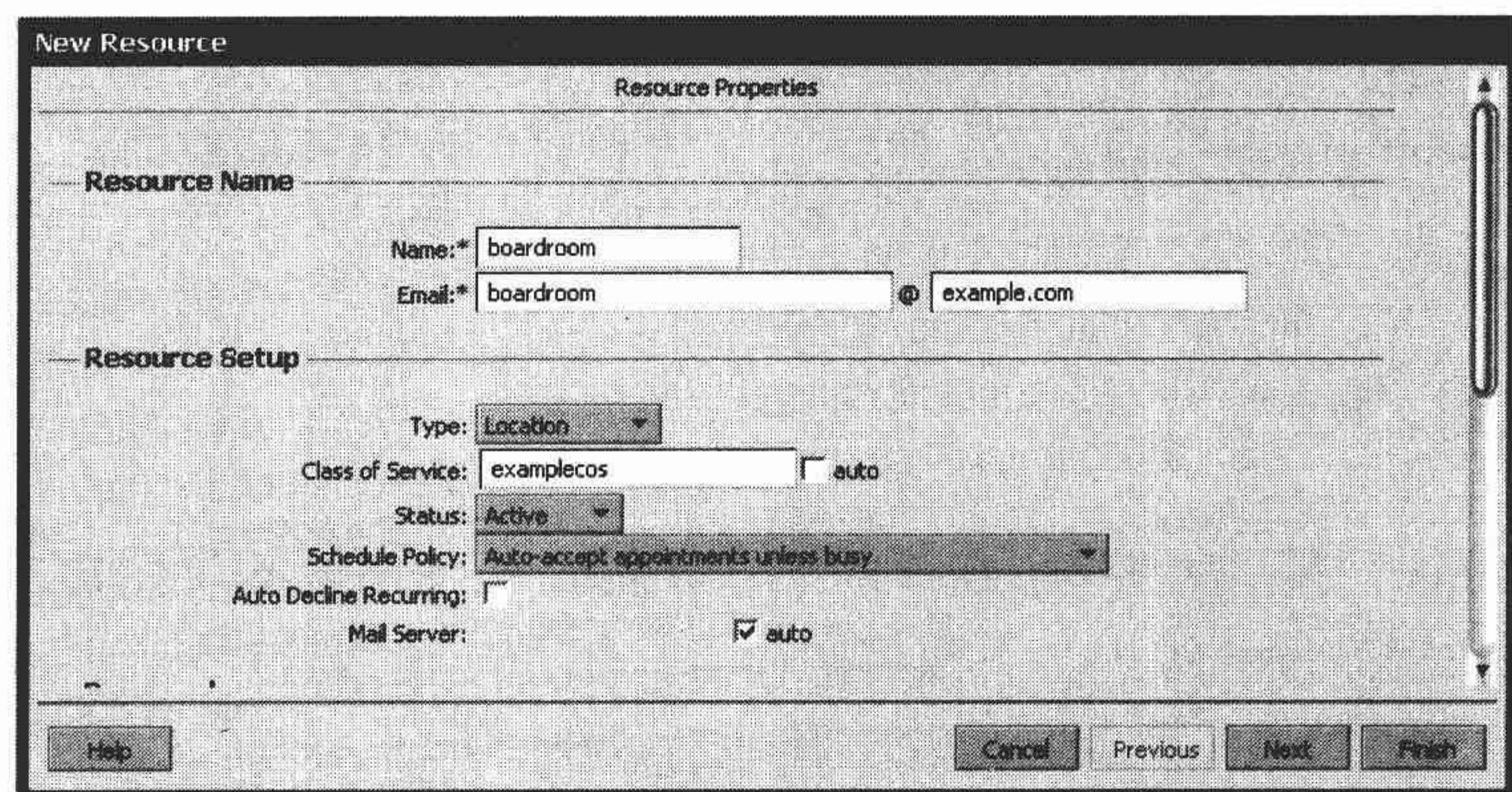


图 15-29 添加一个新的房间资源

该资源被命名为 boardroom，它也有自己的邮箱地址，其资源类型是 Location。这里的另外一个设置项是 Equipment，该设置项共享 examplecos CoS。新资源的资源状态是 Active，调度策略被设置为 Auto-accept appointments unless busy。调度策略的其他选项包括 Accept all appointments、Manually accept 和 Decline appointments，其他用户可以通过选择这些选项来管理该资源的预约批准。

单击“Next”按钮，为该资源添加更多信息。在图 15-30 中，可以看到我们为该资源添加一个联系人姓名及相关信息，也添加了房间自身的信息。

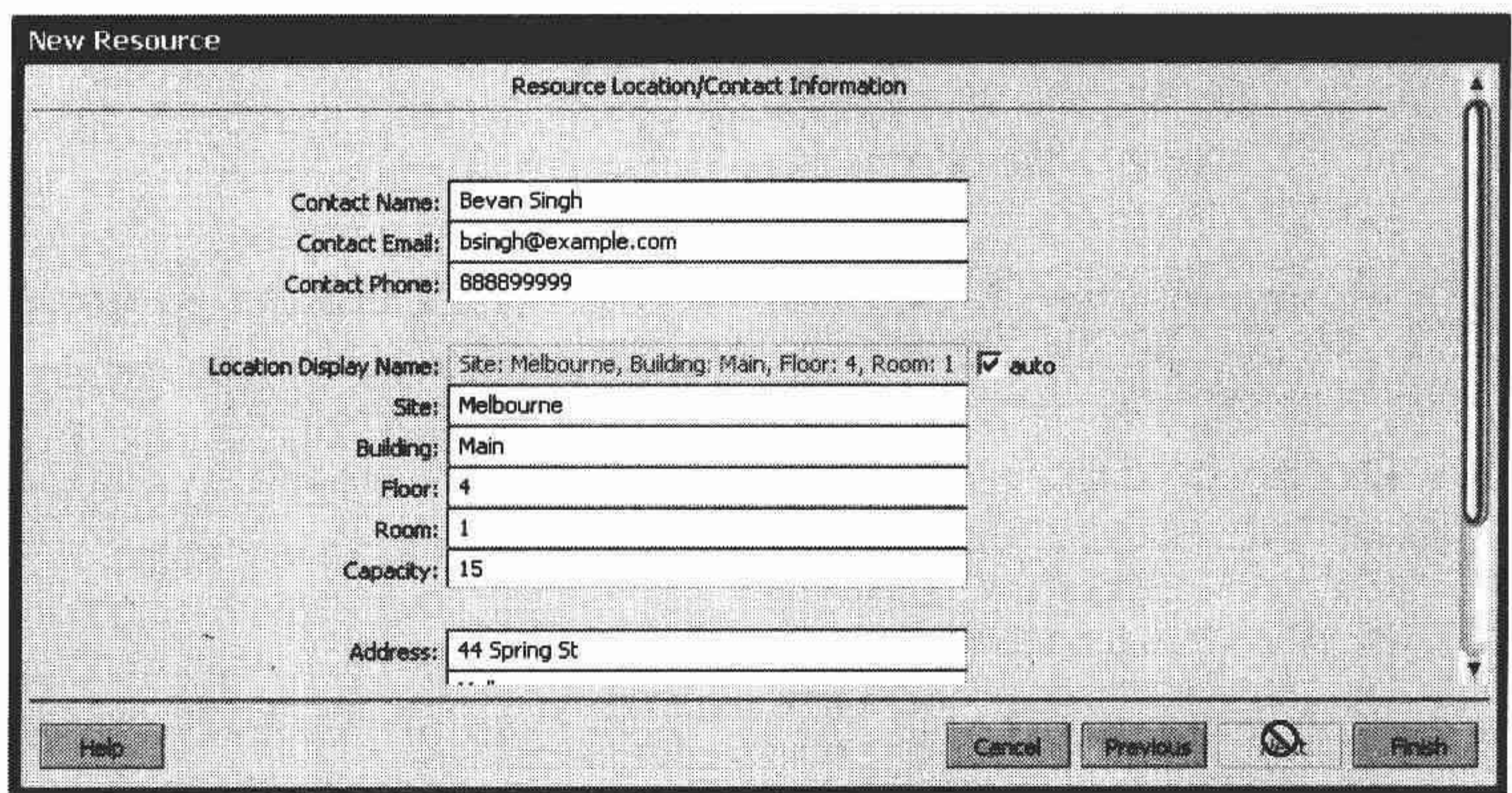
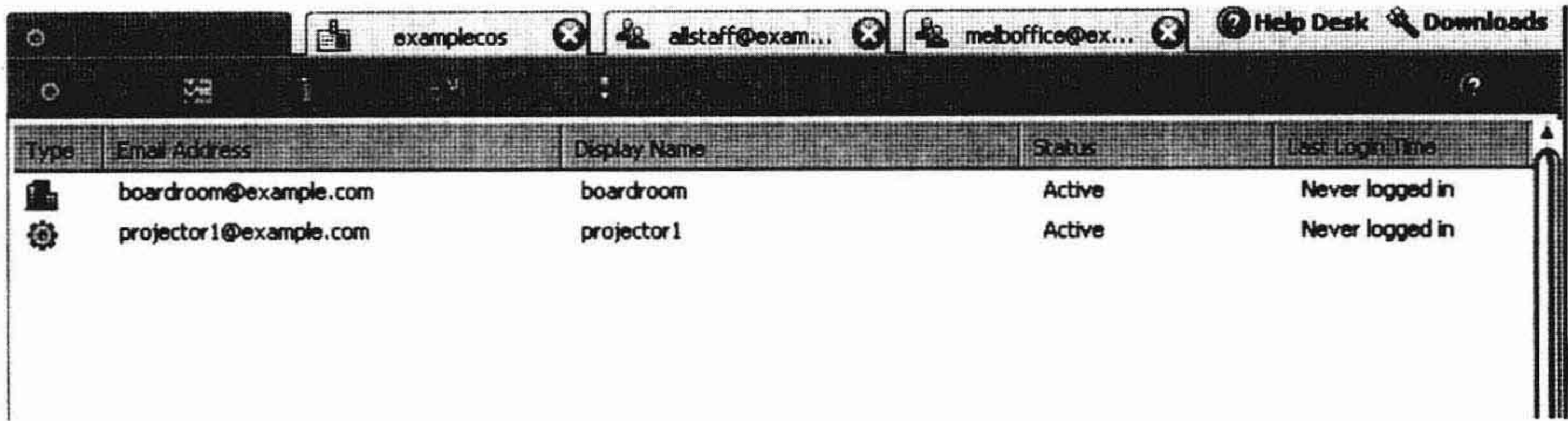


图 15-30 资源 boardroom 的联系人和位置信息

如果对所填写的信息感到满意，单击“Finish”按钮。可以看到已创建好的地点（Location）资源。这里还有一个设备（equipment）资源，它是我们之前采用同样的过程创建的（见图 15-31）。



Type	Email Address	Display Name	Status	Last Login Time
Location	boardroom@example.com	boardroom	Active	Never logged in
Equipment	projector1@example.com	projector1	Active	Never logged in

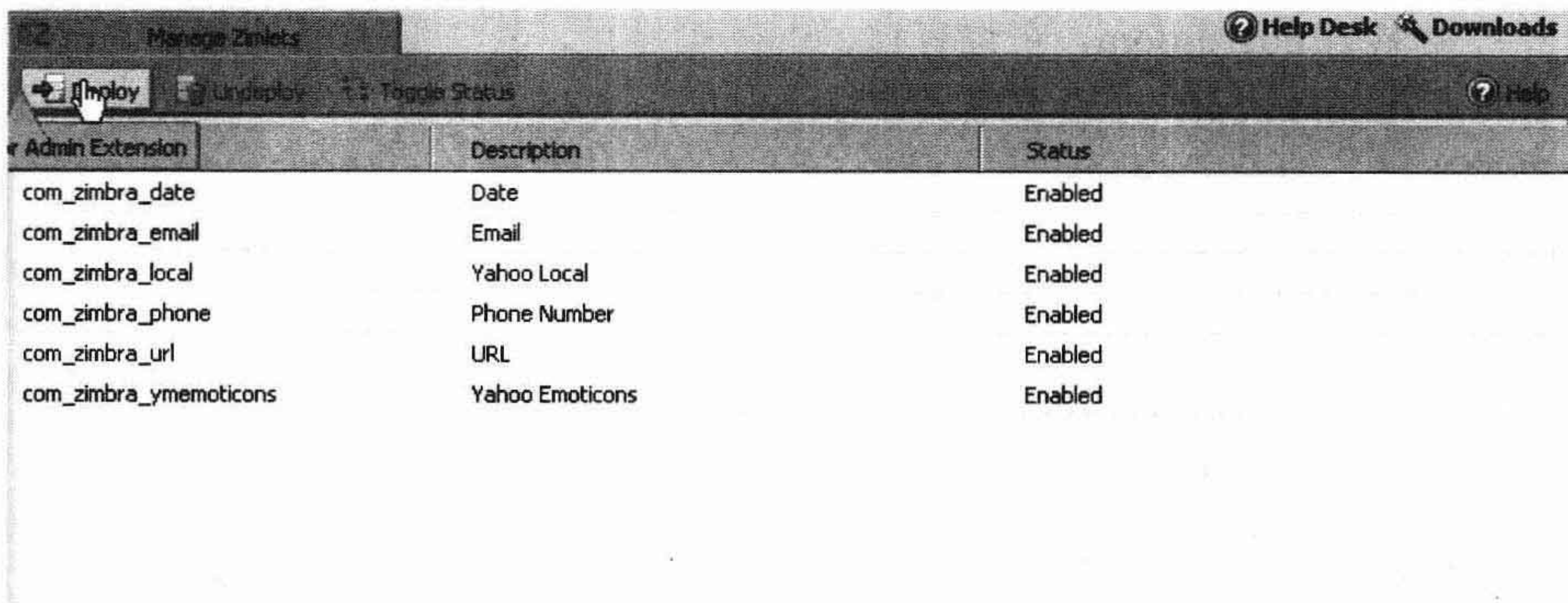
图 15-31 可用资源

现在，已经创建好用户账号、邮箱别名、邮件分发列表和资源，可以开始使用 Zimbra 服务器了。在使用 Zimbra 服务器之前，准备介绍管理控制台的其他方面的一些内容，首先介绍 Zimlet 管理。

15.2.11 添加 Zimlet

已经讲过，Zimlet 是附加的程序或脚本，它为 Zimbra 服务器提供附加功能。Zimlet 通常由 Zimbra 社区用户群开发和维护。下面将向读者介绍如何使用管理控制台添加一个 Zimlet。

在以下 Zimbra 站点可以找到 Zimlet 集合“<http://gallery.zimbra.com/>”。可以把 Zimlet 下载到桌面，然后使用 Zimlets 菜单把它部署到 Zimbra 服务器上。图 15-32 显示的就是“Zimlets”标签窗口。



Admin Extension	Description	Status
com_zimbra_date	Date	Enabled
com_zimbra_email	Email	Enabled
com_zimbra_local	Yahoo Local	Enabled
com_zimbra_phone	Phone Number	Enabled
com_zimbra_url	URL	Enabled
com_zimbra_ymemoticons	Yahoo Emoticons	Enabled

图 15-32 Zimlets 标签窗口

选择部署一个叫做 emailquotes 的 Zimlet。我们已经把文件 com_zimbra_emailquotes 下载到桌面上。现在，单击“Deploy”按钮，浏览目录找到该文件，如图 15-33 所示。

下一步是单击“Deploy”按钮。然后开始安装该 Zimlet 并显示其安装进度。如果一切顺利，在安装结束的时候，会看到一个安装完成信息（见图 15-34），否则，会得到一个错误信息，该信息指明问题所在。

既然已经安装好这个 Zimlet，就可以把它添加到 examplecos CoS，使所创建的所有用户都可以使用该 Zimlet（见图 15-35）。

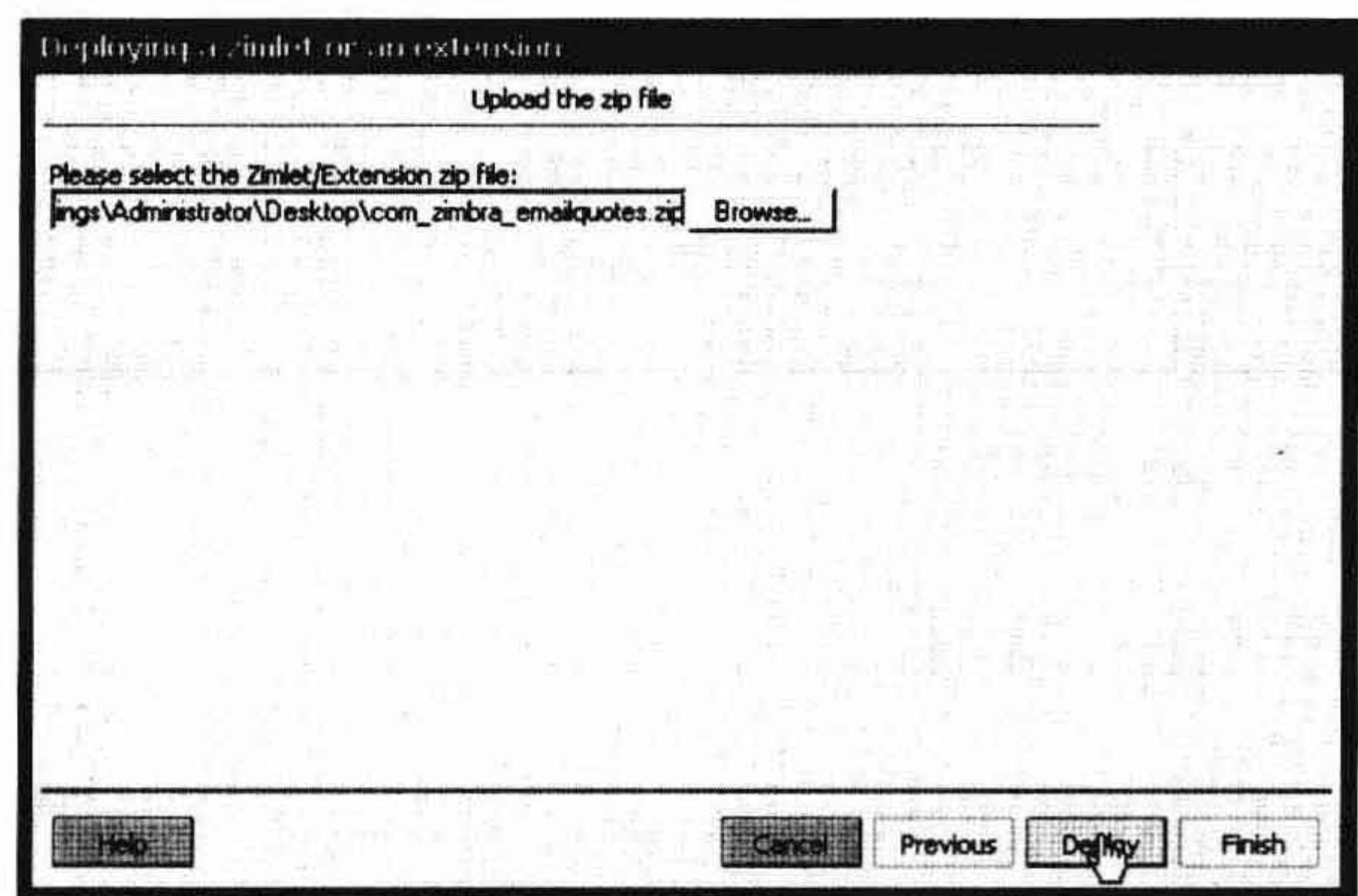


图 15-33 找到一个 Zimlet

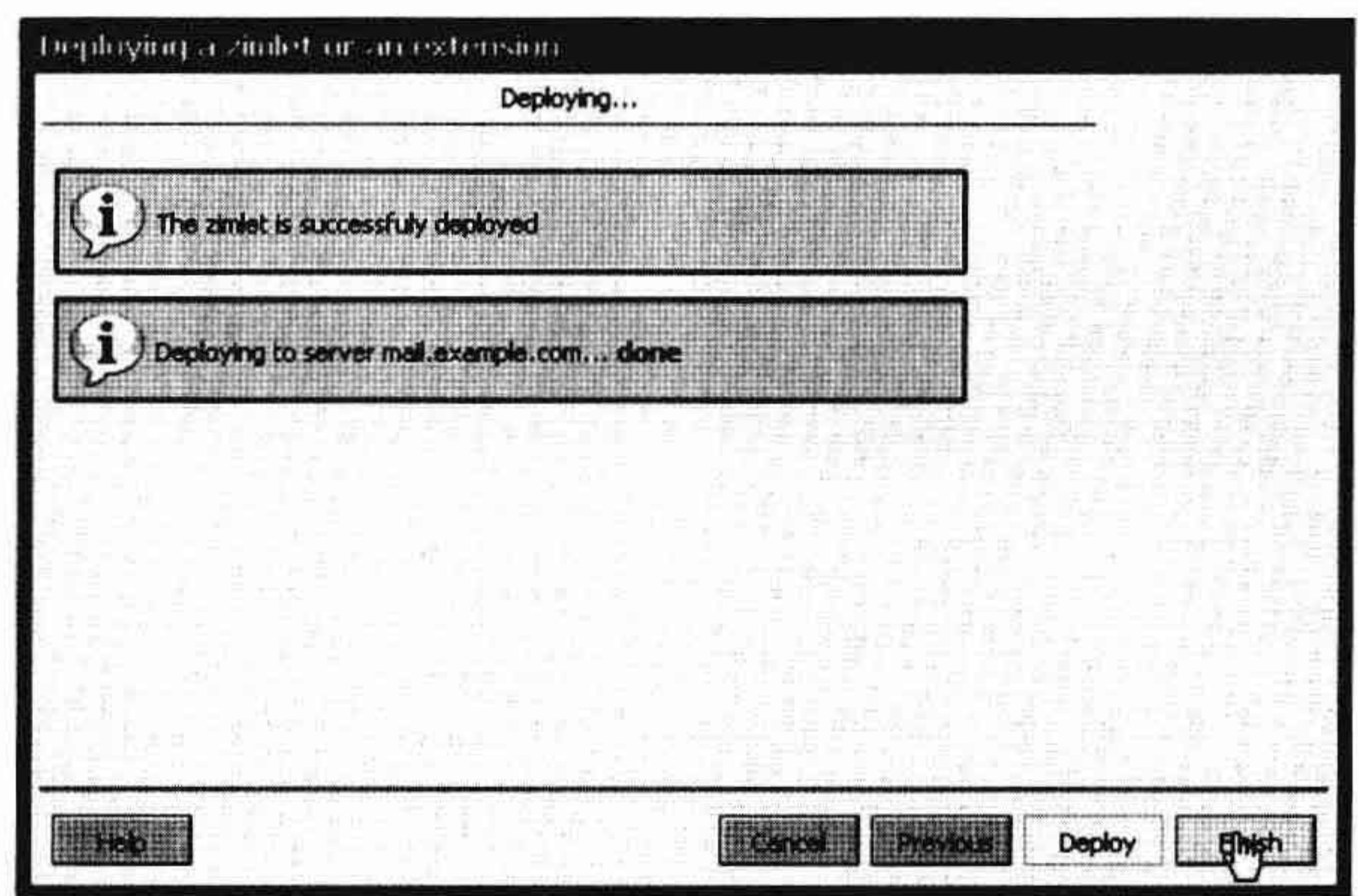


图 15-34 Zimlet 安装完成

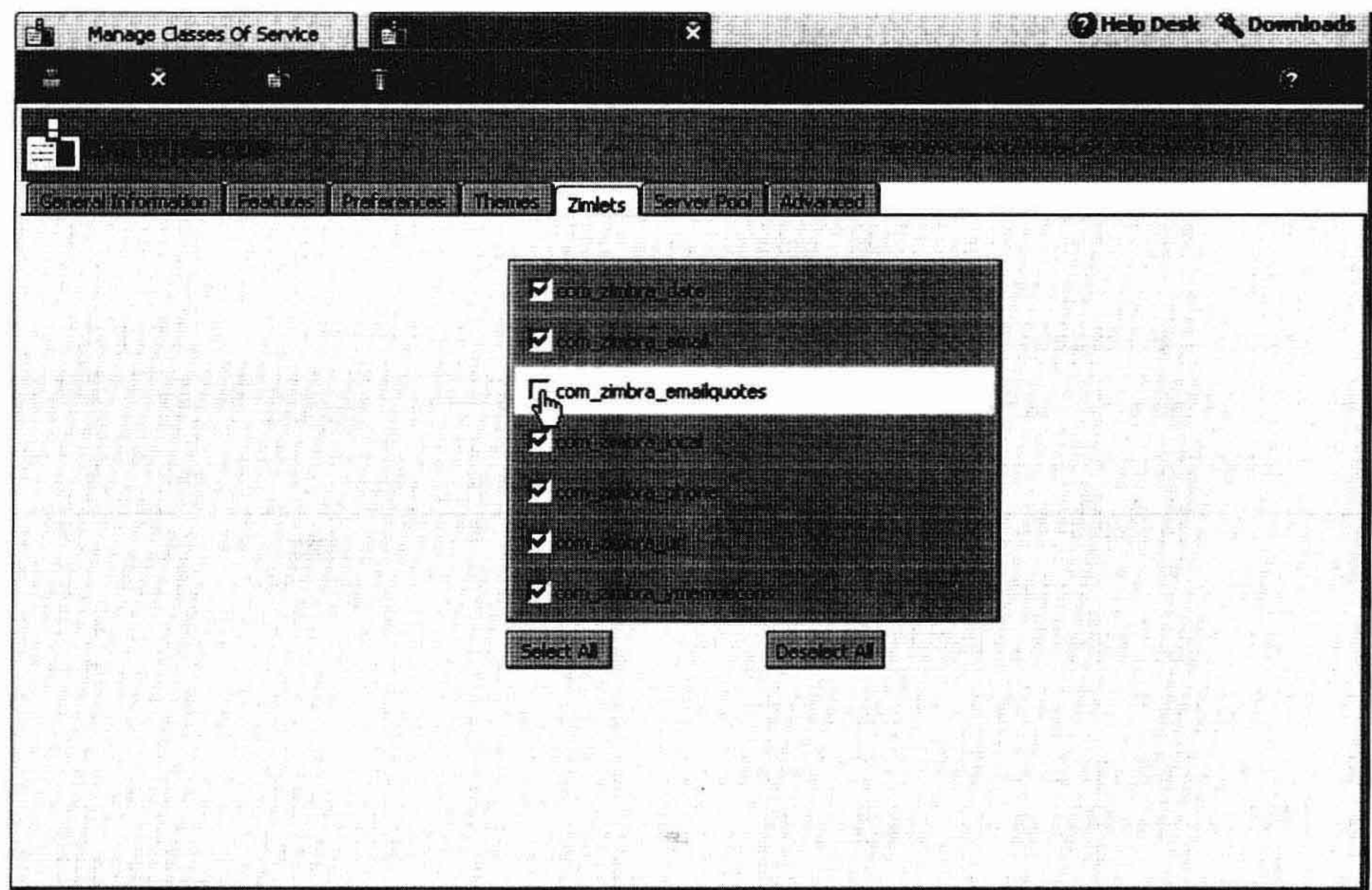


图 15-35 把这个 Zimlet 添加到 examplecos CoS

本章后面的“使用 Zimlet”一节将向读者介绍如何使用 emailquotes Zimlet。

读者还可以用同样的过程添加 Admin Extensions（管理扩展组件）。Admin Extensions 类似于 Zimlet，但它用于管理控制台。之前用过的批量上传工具就是管理扩展组件的一个例子，它可以作为管理扩展组件安装。现在已经介绍完如何添加一个 Zimlet，接下来将向读者介绍如何把自身的 SSL 认证添加到 Zimbra。

15.2.12 添加一个 SSL 认证

无论何时，都应该努力确保通信尽可能的安全。举例而言，我们希望加密 Zimbra 服务器的所有 Web 通信。现在，向读者介绍如何创建一个认证请求，以及如何把该认证安装到 Zimbra 服务器。首先，单击“Manage Certificates”菜单。在图 15-36 中，可以看到下一步是单击“Install Certificate”按钮。

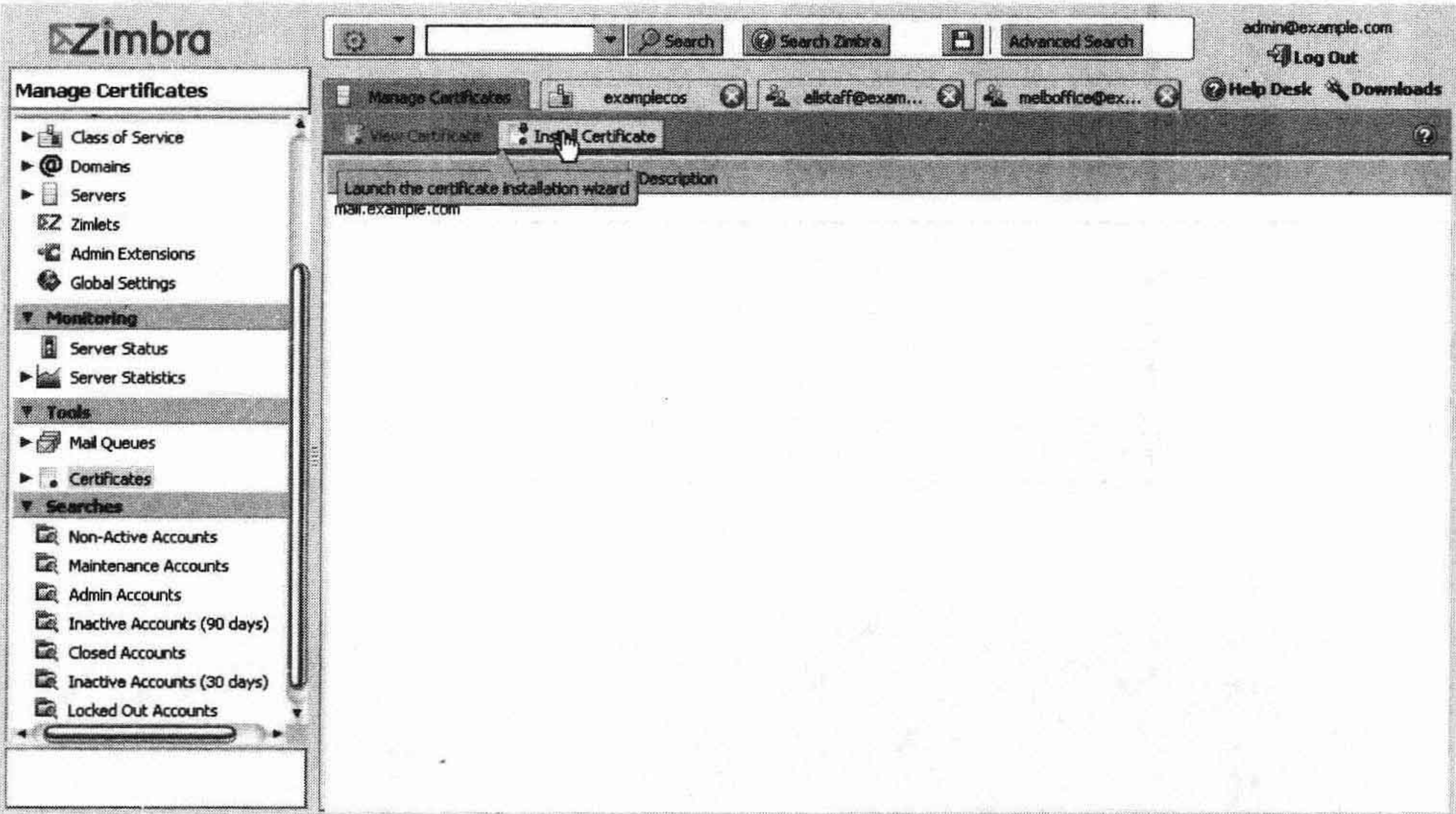


图 15-36 SSL 认证窗口

单击“Install Certificate”按钮后弹出一个如图 15-37 所示的窗口，在该窗口上选择用于安装认证的目标服务器。我们选择唯一可用的选项“mail.example.com”。

接着看到一组安装选项，如图 15-38 所示。该窗口允许安装自签名（self-signed）的认证、创建 CSR 或者安装从商业机构获得的认证。在本例子中，打算创建一个 CSR，这样就可以使用自己在第 10 章创建的认证中心（Certificate Authority）签发它。尽管准备自己签发认证，我们还是选择“Generate the CSR for the commercial certificate authorizer”选项。

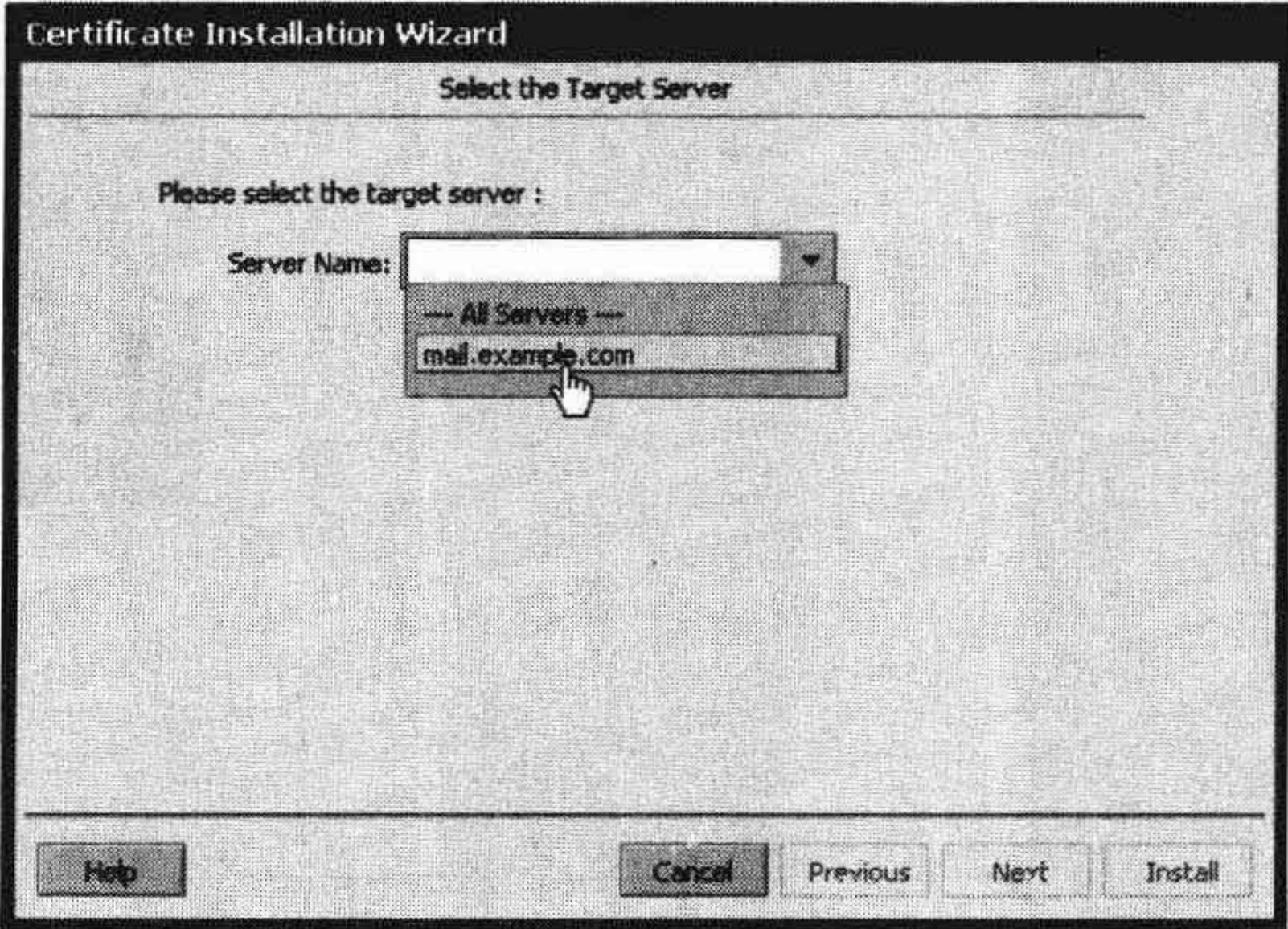


图 15-37 选择目标服务器

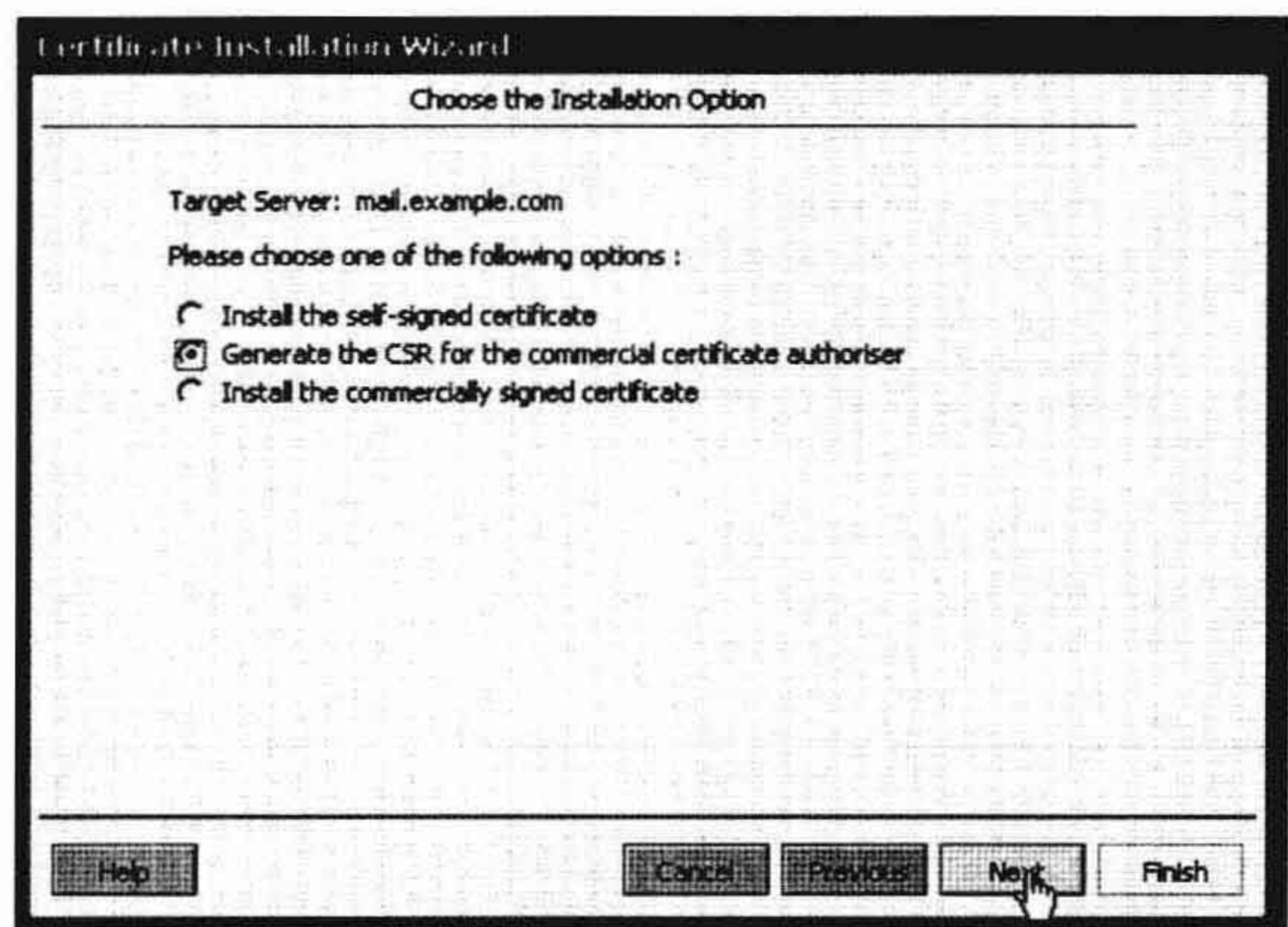


图 15-38 选择生成一个 CSR

现在需要填写生成 CSR 必需的信息。这里要求必填的字段是“Common Name”和“Organisation Name”。“Common Name”是主机名字，“Organisation Name”是认证中心签发认证所必需的，其他信息与主机对应。图 15-39 显示如何在该窗口中填写这些字段。

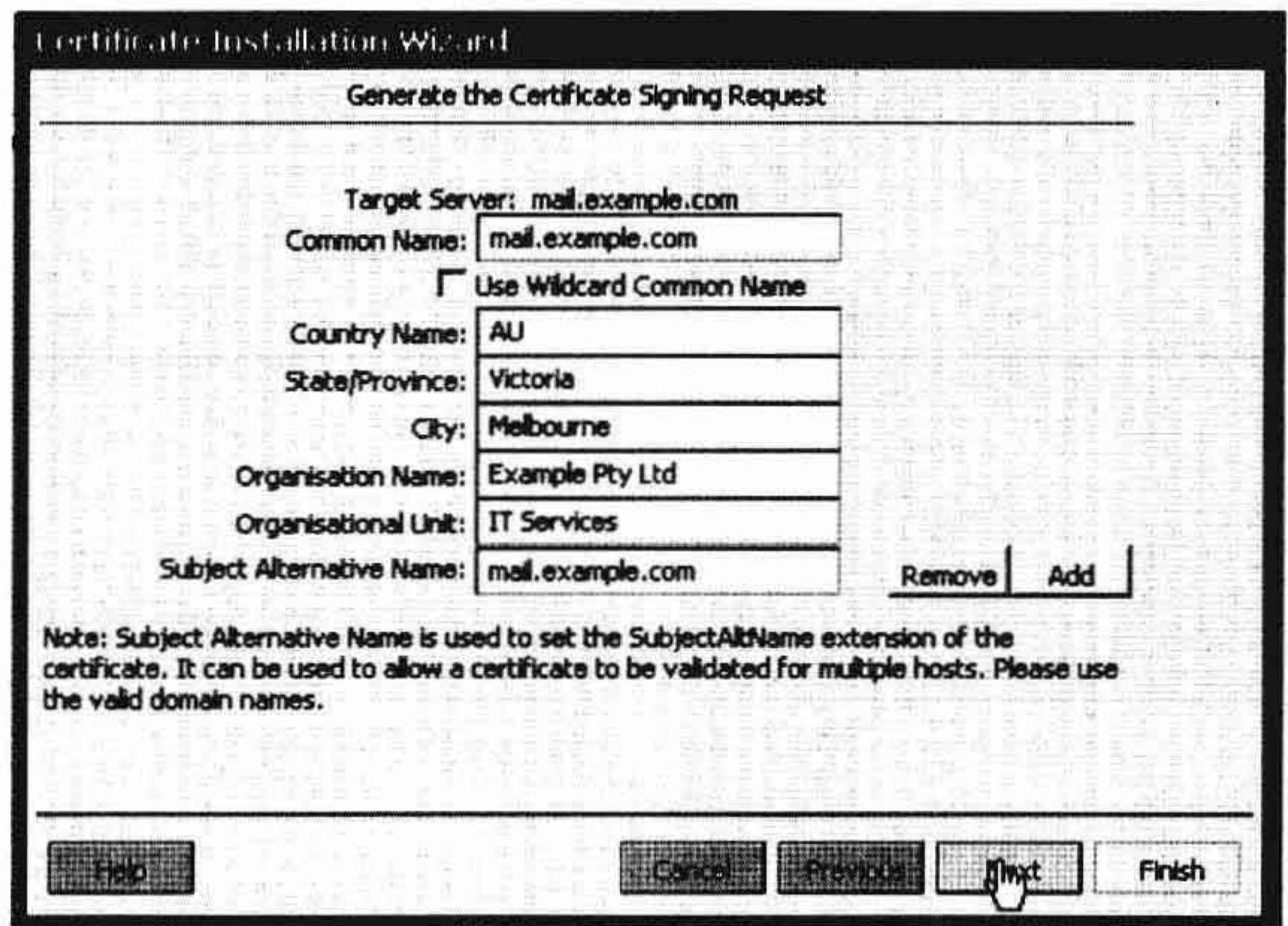


图 15-39 CSR 信息

把 CSR 保存到一个新文件。如果打算通过一个商业机构来签发该 CSR 文件，就要把 CSR 文件发送给该机构。当前情况下，准备按照以下步骤签发自己的认证。首先使用认证中心把认证复制到主机，然后输入以下命令。

```
$ cd /etc/CA
$ sudo openssl ca -out /root/mail.example.com.cert -config ./openssl.cnf
-infiles /root/current.csr
```

接着需要输入 CA（认证中心）的密码，然后签发认证。一旦完成上述步骤，就可以把该认证添加到 Zimbra 邮件服务器。在图 15-40 中，可以看到我们回退到选择安装认证方式的地方。选中“Install Certificate”，并选择目标主机为“mail.example.com”。

现在需要找到认证文件和根 CA 的位置（见图 15-41）。

如图 15-42 所示，现在可以复查该认证的详情。

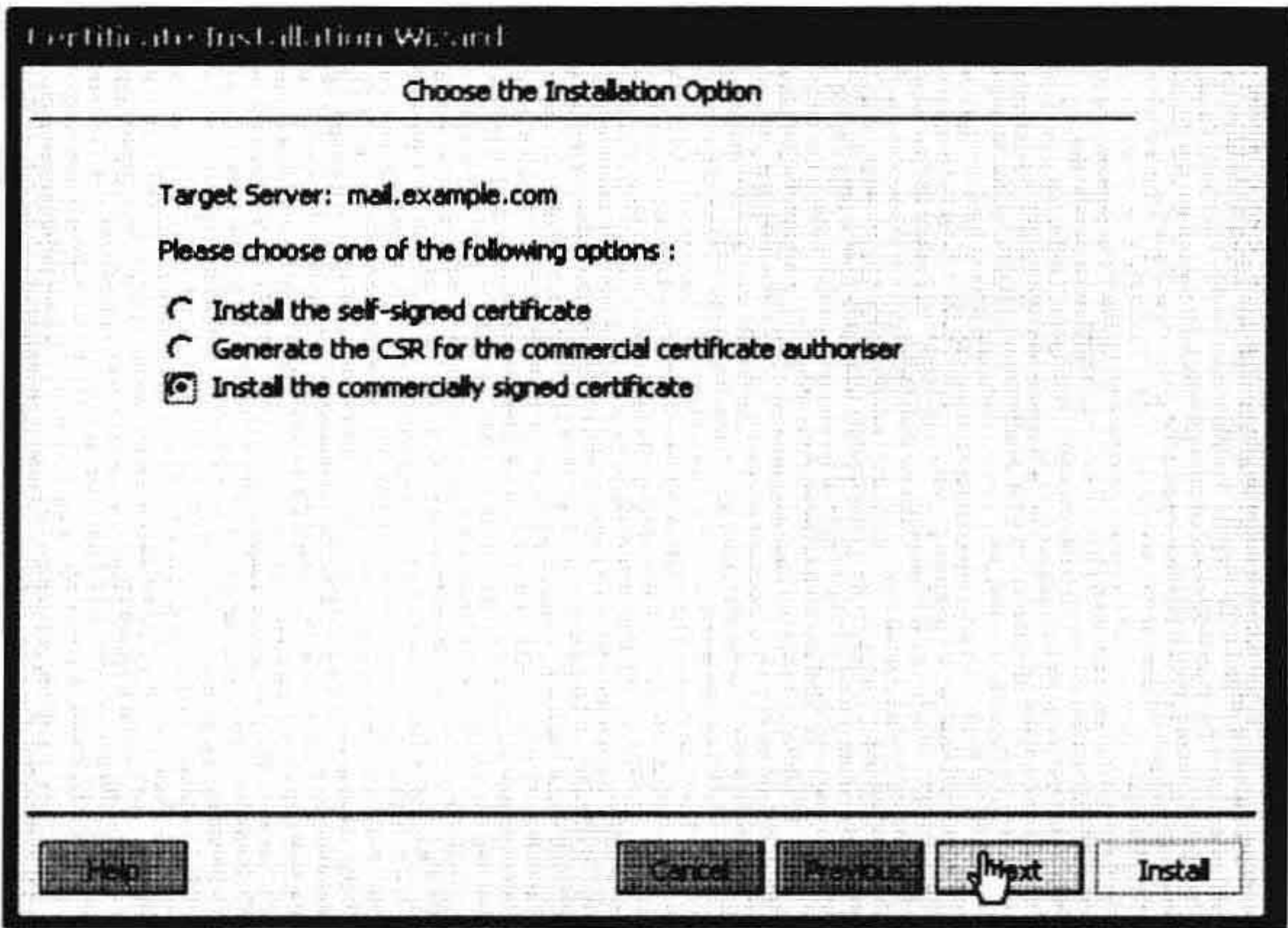


图 15-40 选择安装我们的认证

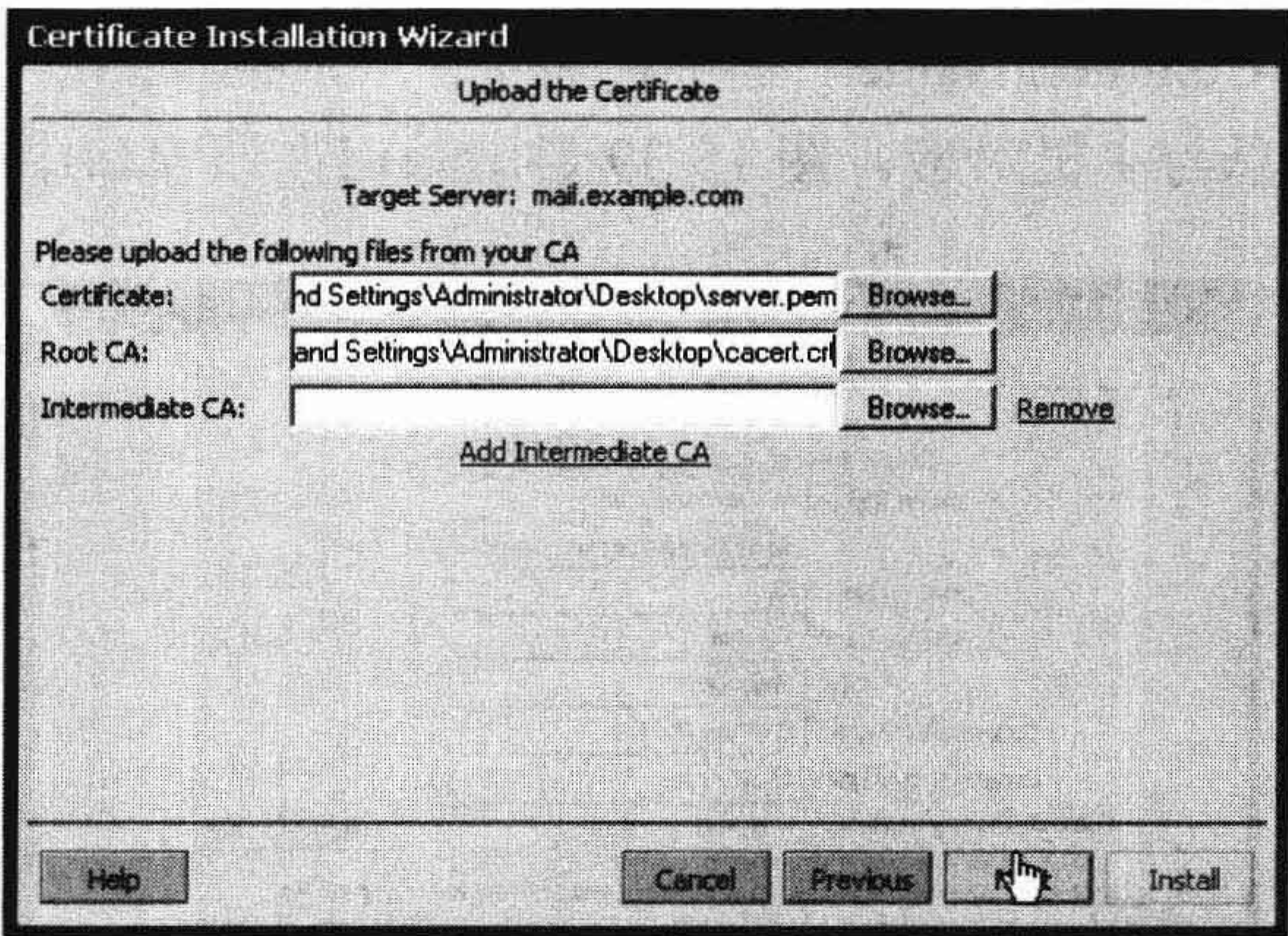


图 15-41 寻找认证文件和根 CA 的位置

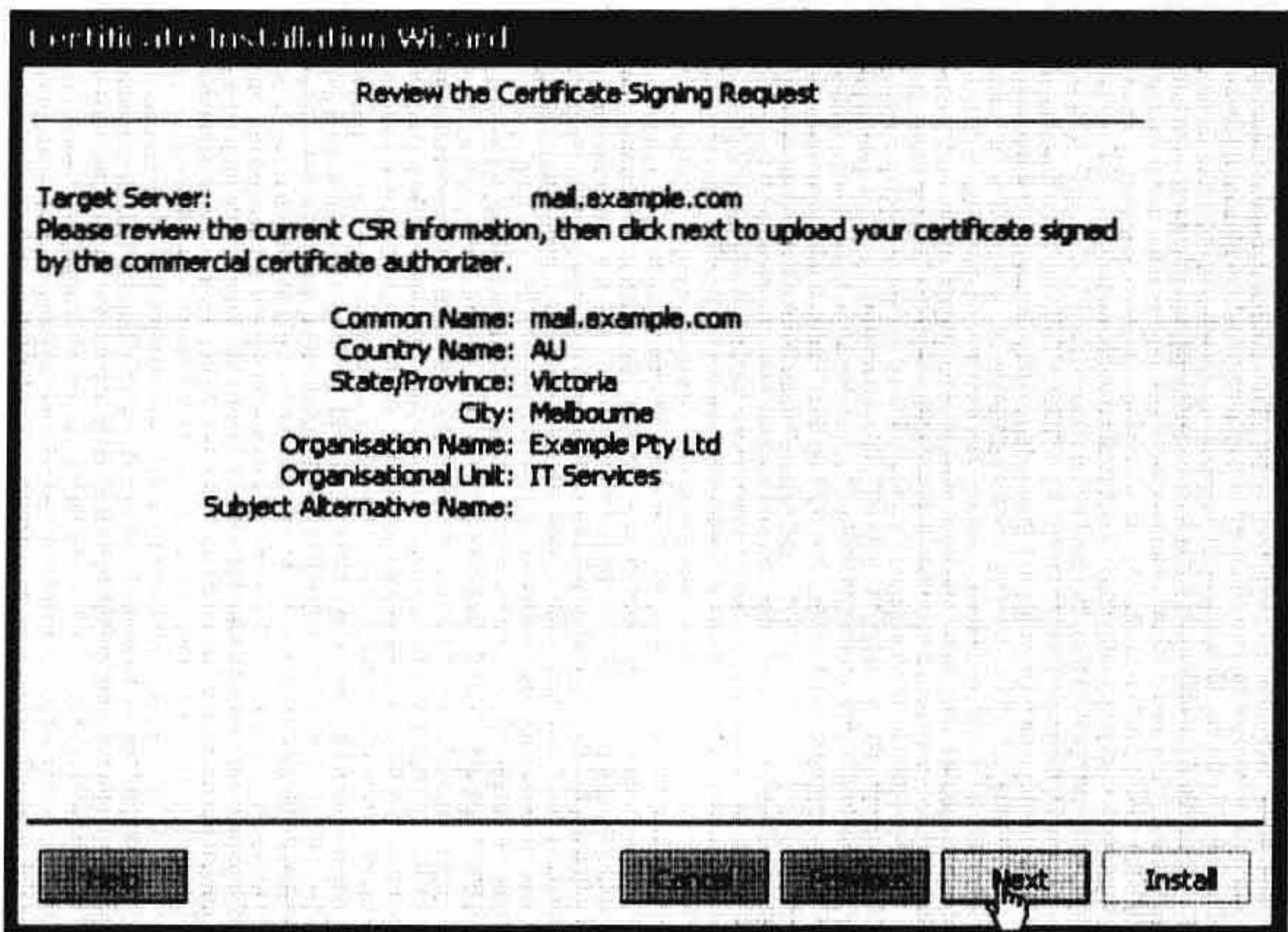


图 15-42 确认认证信息

单击“Next”按钮，弹出安装认证的窗口（见图 15-43）。

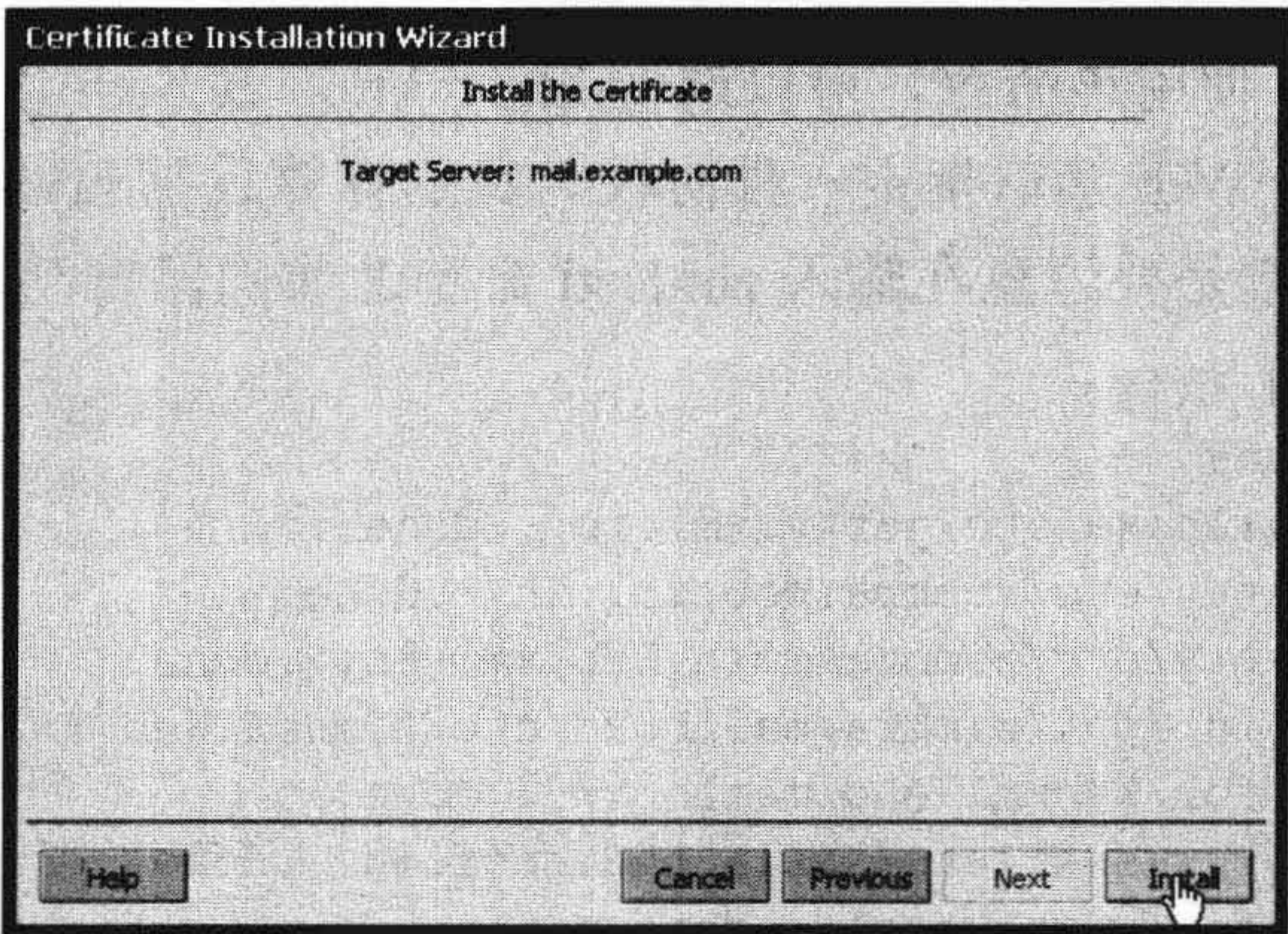


图 15-43 安装认证

接下来，会看到已被正确安装的认证的确认信息（见图 15-44）。

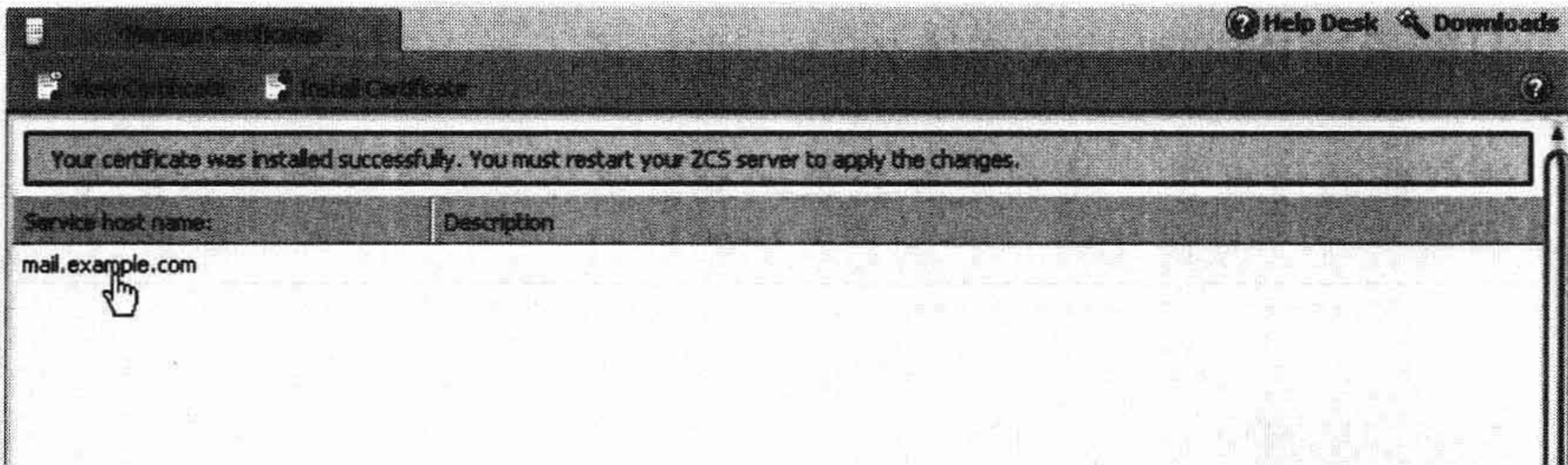


图 15-44 认证安装成功

最后，可以查看刚刚安装好的认证信息，高亮选中该项认证，然后单击“View Certificate”按钮。读者可以从图 15-45 中看到认证信息的结果显示。

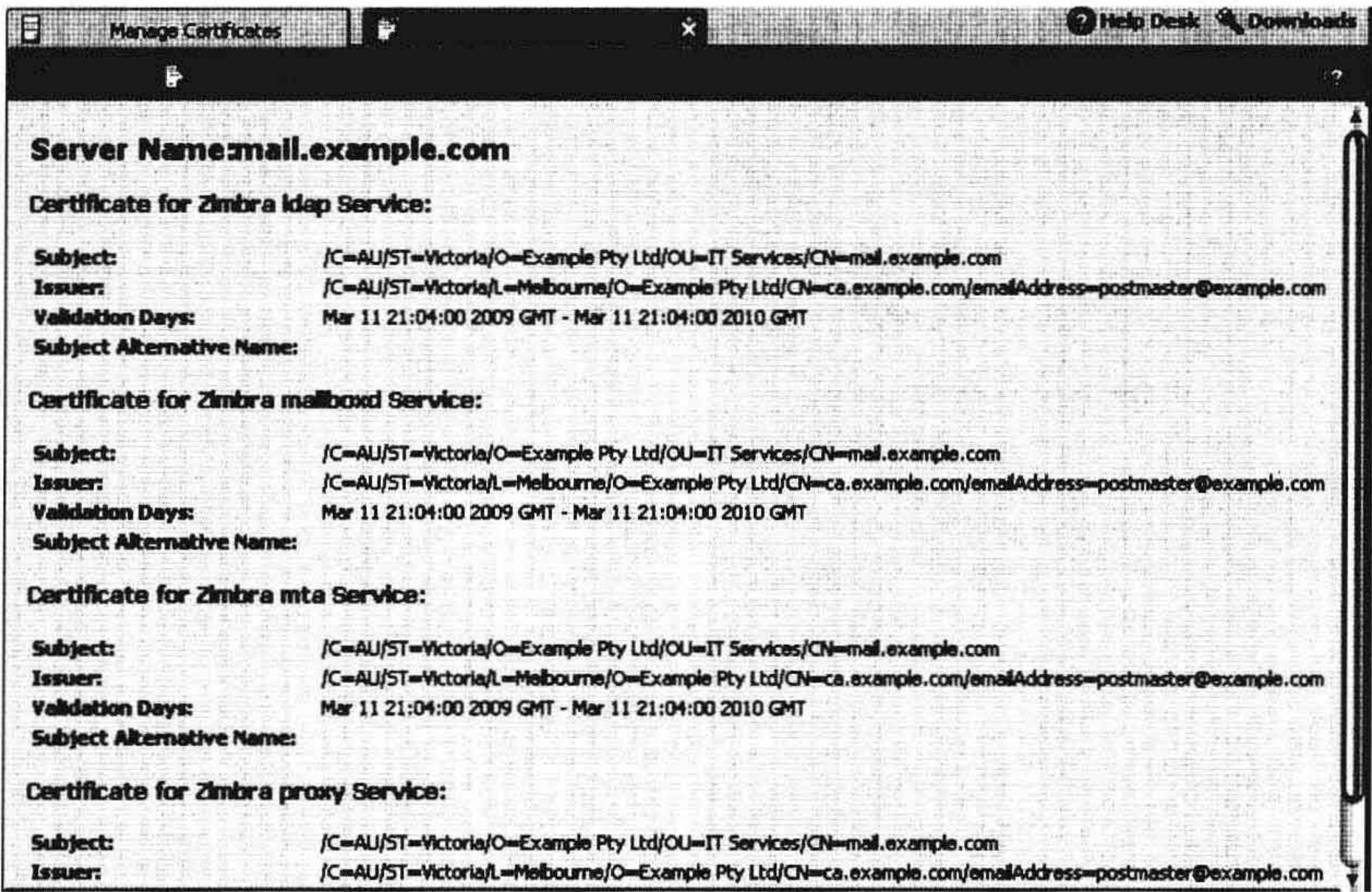


图 15-45 查看安装好的认证

认证现在已经安装完毕，下面需要使用 Zimbra 的命令 `zmtlsctl`（该命令管理 HTTPS 的使用）发起 HTTPS 会话。使用该命令，可以选择 `http`、`https` 或 `both` 连接方法中的一种，以便为连接 Zimbra 服务器的 Web 接口指定一种协议。我们打算只用 `https`，因此在邮件服务器的命令行输入下列命令，但是要注意在输入 `zmtlsctl` 命令之前确保自己的用户身份是 `zimbra`。

```
zimbra@mail:~$ zmtlsctl https
Setting tls mode to https
Updating /opt/zimbra/mailboxd/etc/jetty.xml.in...done.
Updating /opt/zimbra/jetty/etc/zimbra.Web.xml.in...done.
Updating /opt/zimbra/jetty/etc/zimbraAdmin.Web.xml.in...done.
Updating PROTOCOL MODE in /opt/zimbra/mailboxd/etc/zimbra.Web.xml.in...done.
Rewriting config files for Webxml and mailboxd...done.
Updating /opt/zimbra/cyrus-sasl/etc/saslauthd.conf.in...done.
Rewriting config files for cyrus-sasl...done.
Setting ldap config zimbraMailMode https for mail.example.com...done.
zimbra@mail:~$ zmcontrol stop
zimbra@mail:~$ zmcontrol start
```

可以看到，这里把 `https` 服务设置为 TLS 模式，然后用 `zmcontrol` 命令先后停止和启动该服务。现在，只能通过 HTTPS 协议连接 Zimbra 主机了。

提示：在任何可能的情况下，特别是在登录网站需要用户名和密码验证的情况下，我们应该尽可能地使用最高安全性的协议，这就是本例选择 HTTPS 协议而不是 HTTP 协议的理由。

15.2.13 全局设置

在管理控制台的“Manage Global Settings（管理全局设置）”中，可以查看和管理 Zimbra 服务器的进一步设置。我们可以在此定义可被服务器接收的附件、Postfix 服务器的 MTA 设置以及 IMAP/POP、反病毒和垃圾邮件的设置。以下是对服务器所做的改动，这些改动对读者同样重要。

图 15-46 显示的是一般信息设置（General Information settings）。可以看到，全局地址列

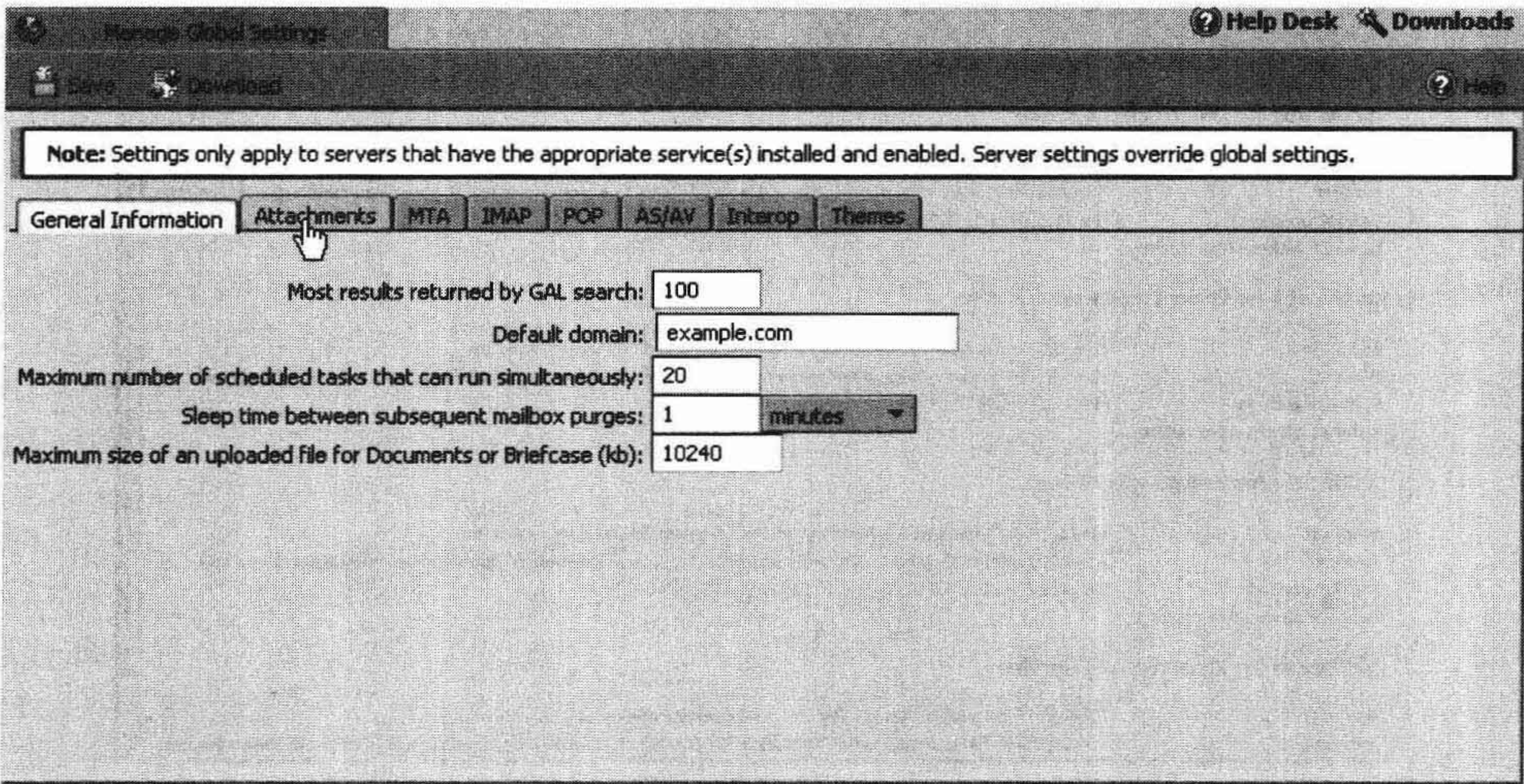


图 15-46 管理控制台“Manage Global Settings”中的一般信息标签页

表 (global address list , GAL) 返回结果的数目上限被设置为 100, 这是默认值。如果读者的公司是一个大公司或者 LDAP 服务器的性能有可能出现问题, 则可能需要把这个数值设置得高一些。还有, 注意能够上传到公文包的文件大小是有限制的 (公文包是可以放置文件的共享用户空间), 这会允许 Zimbra 用户远程访问公文包中的文件。

下一步, 在 “Attachments” 标签页上设置所要接收的附件, 如图 15-47 所示。这里排除以特定扩展名结尾的附件, 例如以 .exe 或 .js 结尾的附件。这样设置有助于保护用户免遭恶意附件的骚扰。我们还可以重置显示附件的 CoS 设置。图 15-47 显示所有可能被用到的文件扩展名。使用标签页底部的 “New extension field” 按钮还可以添加任何需要的扩展名。

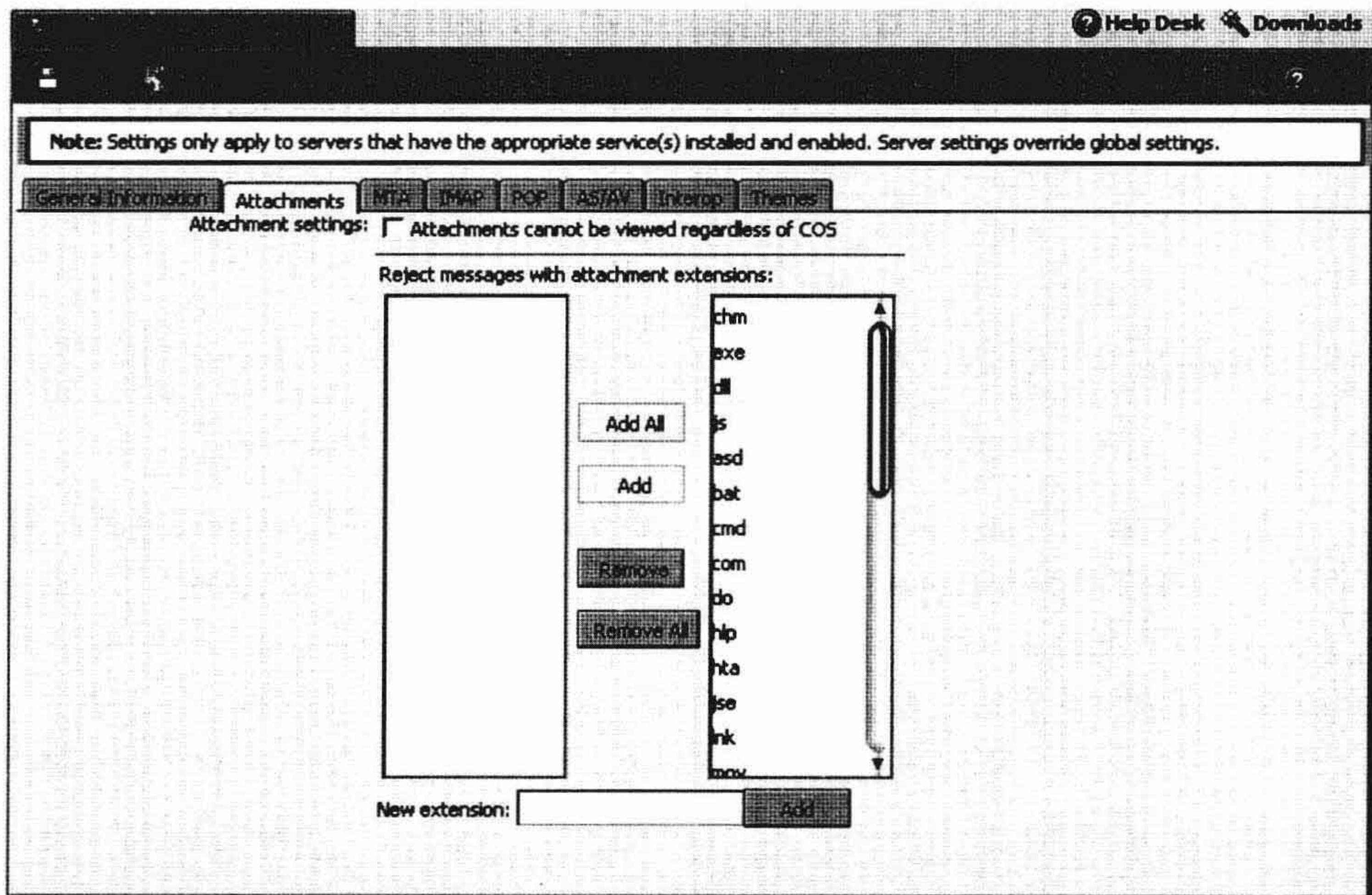


图 15-47 管理控制台 “Manage Global Settings” 中的 “Attachments” 标签页

接下来在 “MTA” 标签页上设置 Postfix 服务器的配置信息。这里是进行诸如指定一个中转 MTA 之类的设置的地方, 指定中转 MTA 使我们可以借助一个边缘 MTA 发送所有邮件。边缘 MTA 把内部的邮件服务器与外部的直接访问隔离开来, 并且还可以做初步的邮件扫描以保护重要的邮件服务器免遭恶意攻击。

在该标签页上, 还可以设置邮件大小的最大值及其他的 MTA 设置项。可以在图 15-48 中看到我们设置的一些设置项。

在 “IMAP” 和 “POP” 标签页上, 我们已经关闭了 IMAP 和 POP 服务。请读者回顾我们在第 10 章介绍的如何配置 POP 和 IMAP, 以及其他用于诸如 Microsoft Outlook 或 Thunderbird 之类电子邮件客户端的协议, 从而连接邮件服务器并获取邮件。

因为打算让所有用户通过 Web 方式访问服务器, 所以我们决定不再启动这些服务。如果读者观察一下这些标签页, 就会发现上面有两个简单的 “Enable IMAP services” 和 “Enable POP services” 复选框, 它们用于开启或关闭相应的服务。在标签页上进行的任何改动必须在重启服务器后才能生效。

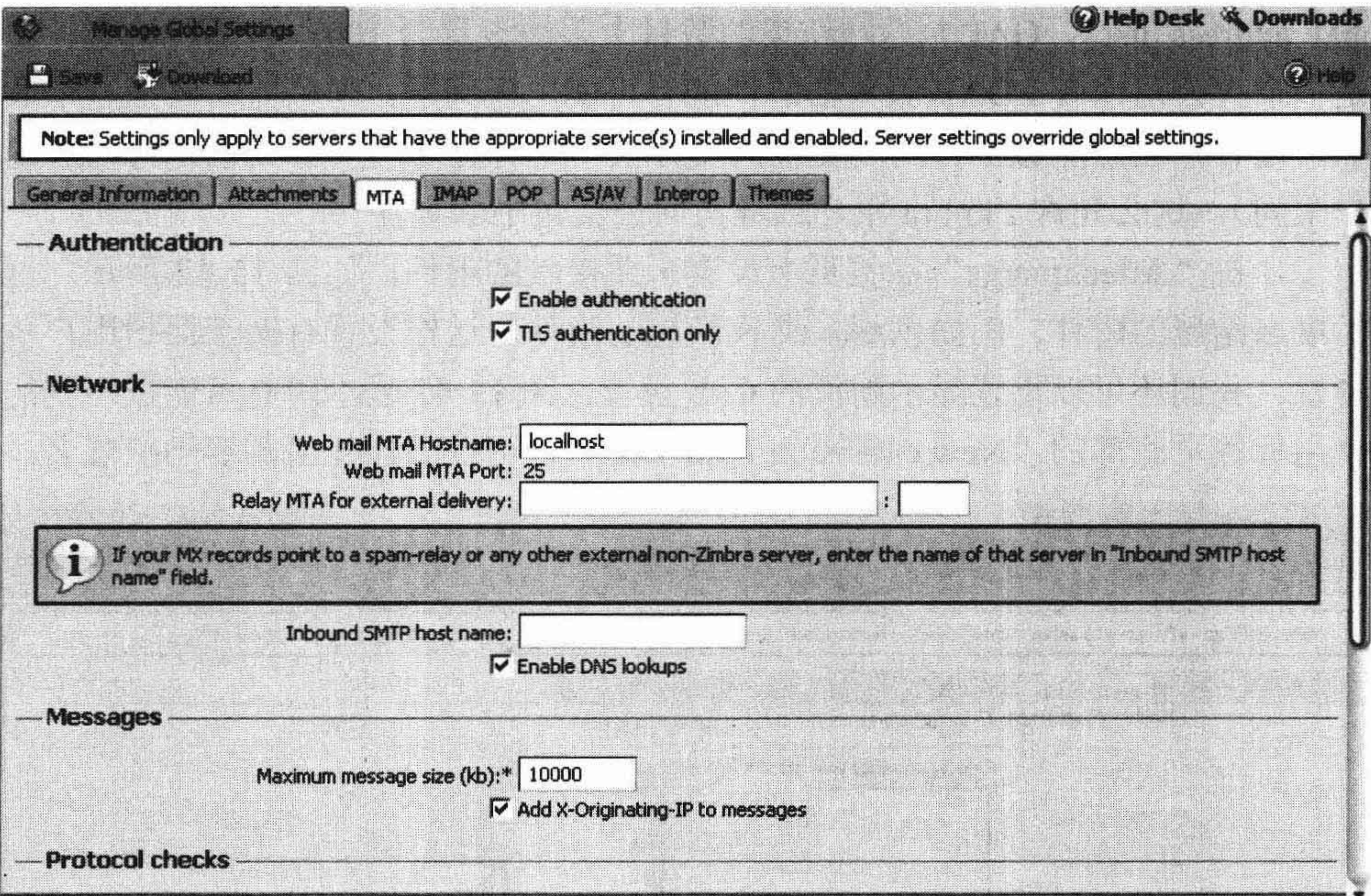


图 15-48 管理控制台 “Manage Global Settings” 中的 “MTA” 标签页

接下来是 “AS/AV” 标签。AS/AV 是 antispam/antivirus（反垃圾邮件/反病毒）的简称。在图 15-49 中，我们把 “Subject prefix” 一项设置为 “SPAM:”，同时还指定了 “Kill percent” 和 “Tag percent” 字段的百分率。垃圾邮件过滤器依据一个可信度来评估可能的垃圾邮件，如果收到的邮件有 75% 的可信度被认为是垃圾邮件，垃圾邮件过滤器将拒绝该邮件，并把它移到邮件服务器的隔离区。如果垃圾邮件过滤器有 33% 的可信度认为该邮件是垃圾邮件，则

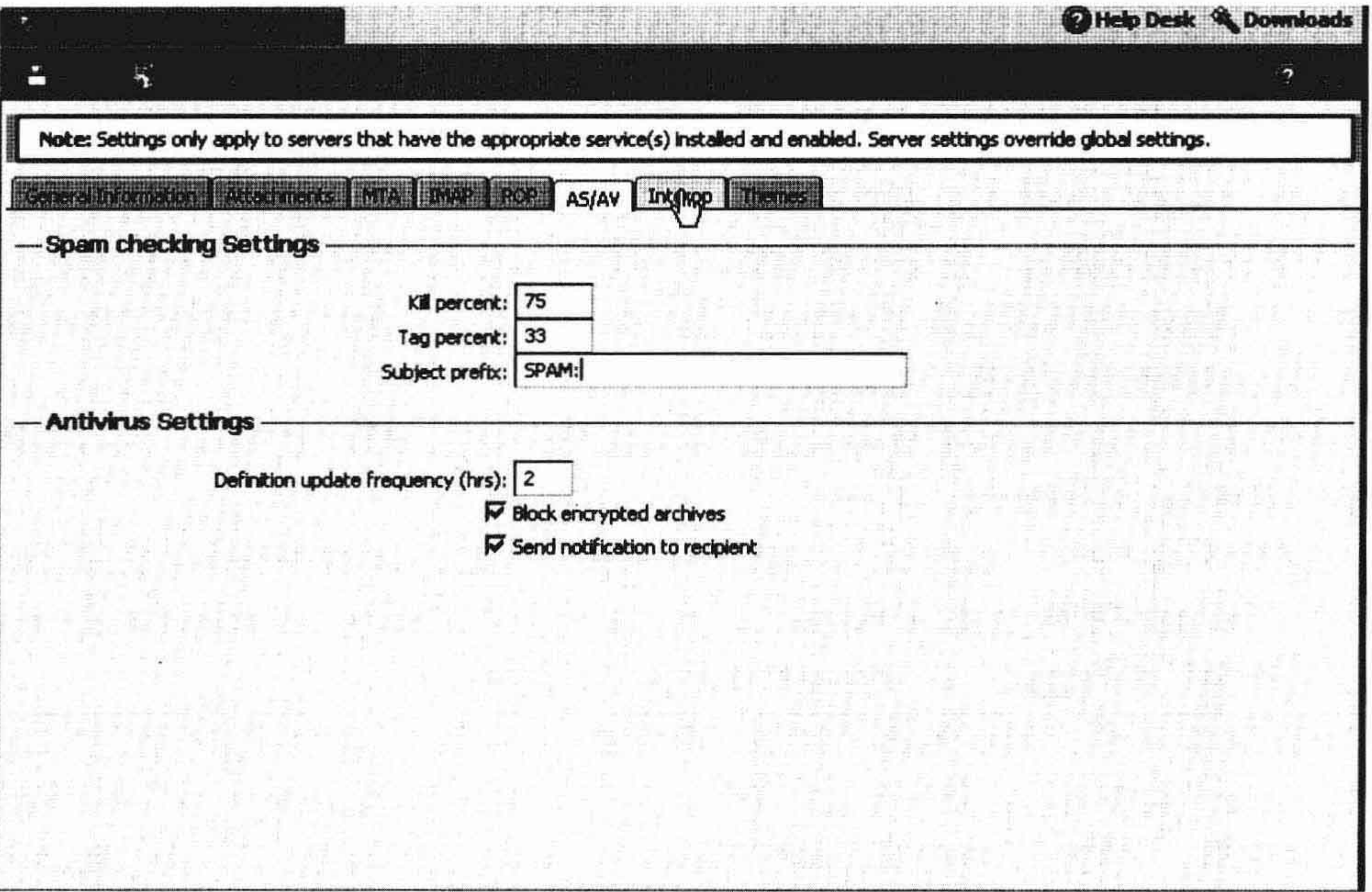


图 15-49 管理控制台 “Manage Global Settings” 中的 “AS/AV” 标签页

过滤器允许该邮件发往预定的接收者，但是会使用标题前缀“SPAM:”标记该邮件，然后交由收信人判断该邮件是否为垃圾邮件。

我们可能处在这样一种情况下：公司网络中现有一台 Microsoft Exchange 服务器，并有一群用户正在使用它。“Interop”标签页允许对这台 Microsoft Exchange 服务器做一些设置。在这种情形下，Microsoft Exchange 服务器用户和 Zimbra 服务器用户可以通过交换“闲/忙”信息，一起安排活动日程。

“Manage Global Settings”的最后一个标签页是“Themes”。该标签页上的设置项允许我们更改前景、背景等地方的颜色。

需要注意的是，在“Manage Global Settings”中所做的改动会被 Servers 中的设置所覆盖。此外，“Manage Global Settings”中的一些设置需要重启 Zimbra 服务器才能生效。Zimbra 服务器会在管理控制台上提供简单易懂的提示信息，建议读者阅读这些信息以获知更多细节。

15.2.14 监控 Zimbra

Zimbra 管理控制台具备监控邮件服务器相关信息的功能。如图 15-50 所示，“Server Statistics”窗口显示各种供用户查看的统计信息。通过该窗口，读者可以查看某段预设时间内的报文数目、报文占用空间和反垃圾邮件/反病毒活动。

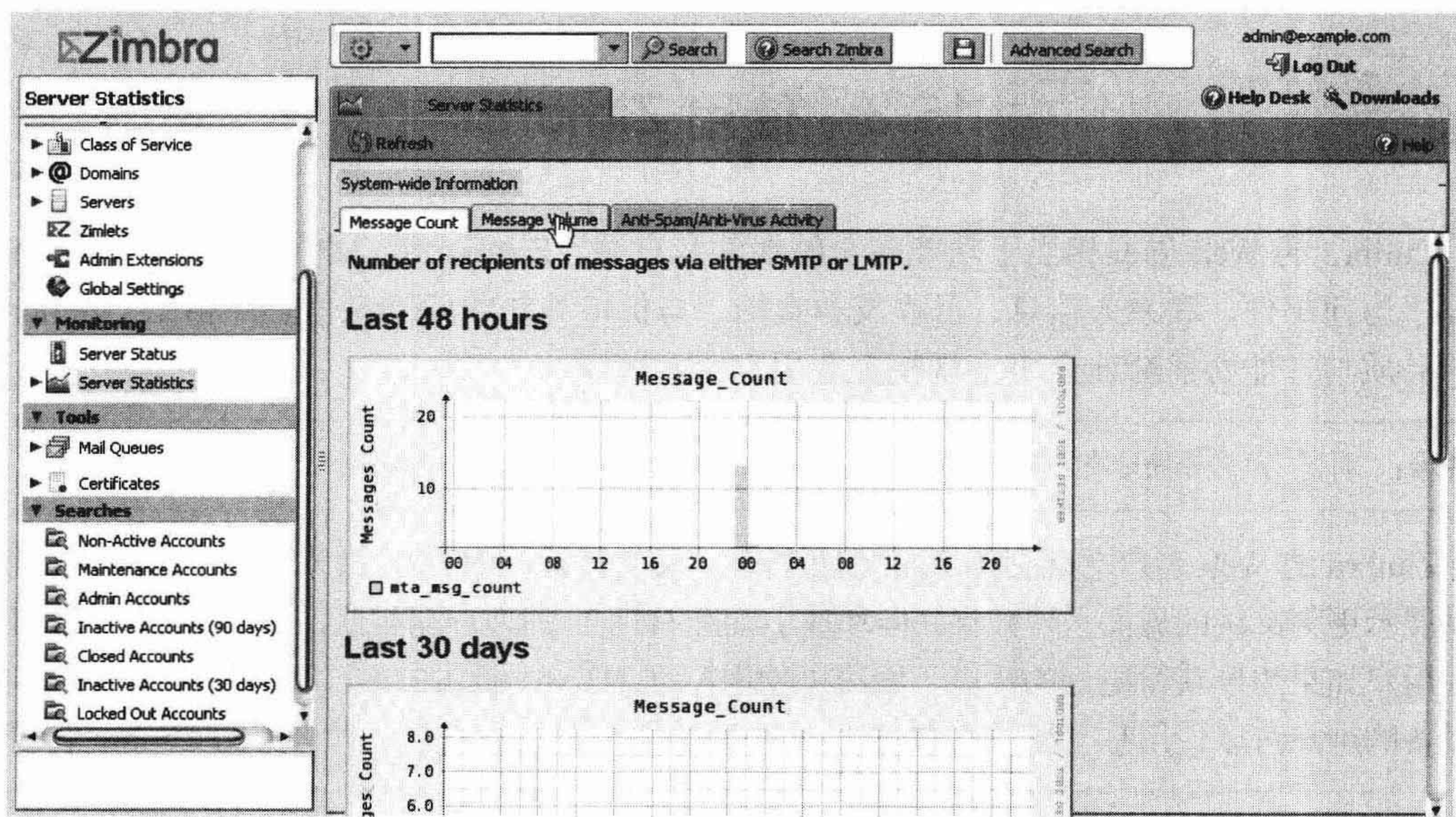
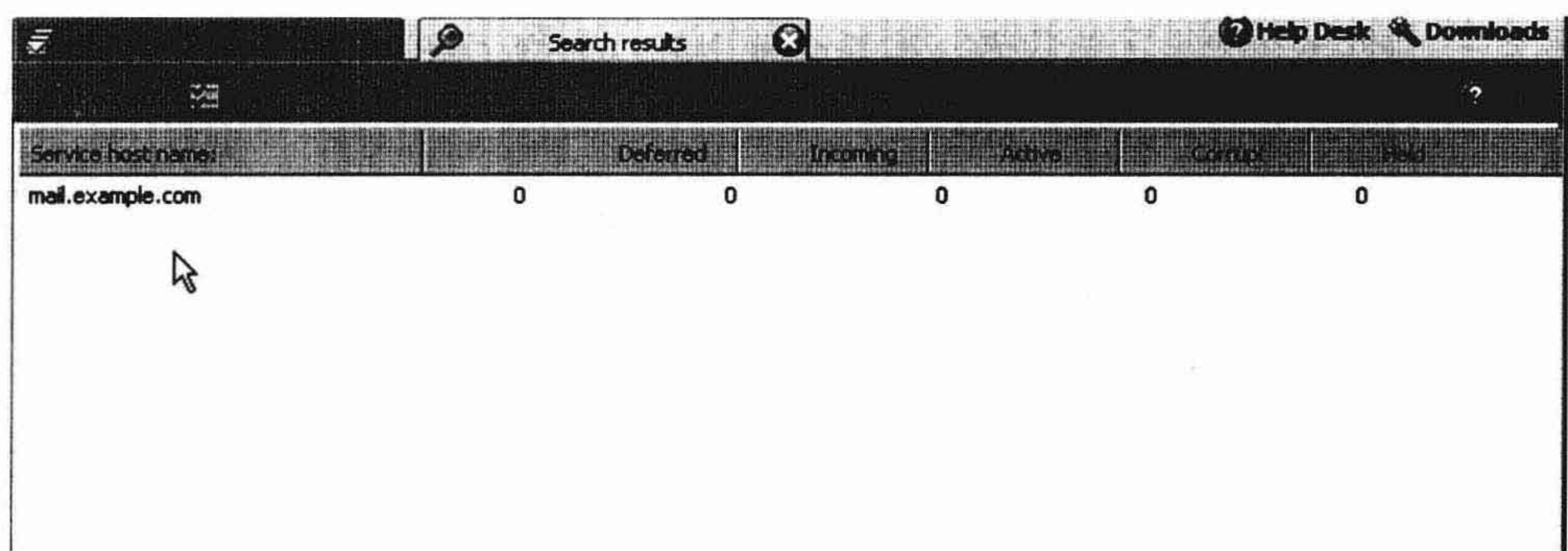


图 15-50 “Server Statistics” 窗口

该窗口还允许用户观察邮件队列（mail queue）的状况。邮件队列表明邮件服务器主机的实际性能及其对可能收到的任何骚扰报文的清理能力。骚扰报文是一种回复报文，源自垃圾邮件或者某些人尝试发送大型邮件的报文。图 15-51 显示处于 Deferred、Incoming、Active、Corrupt 和 Held 等状态的多个邮件队列，可以在 Postfix 服务器上找到这些邮件队列。某个邮件队列的高计数值意味着邮件服务器出现了问题，此时就应该对此问题做进一步调查。



Service host names	Deferred	Incoming	Active	Queue	Held
mail.example.com	0	0	0	0	0

图 15-51 邮件队列

■注：第 10 章讨论过 Postfix 邮件服务器和邮件队列。

至此，关于 Zimbra 服务器管理控制台的介绍全部结束。

■注：除了 ZCS Administrator’s Guide，还有一个关于 Zimbra 配置的优秀资料可以在以下网页中找到，见 “http://wiki.zimbra.com/index.php?title=Main_Page”。

下面探讨如何使用 Zimbra 提供的 Web 界面。

15.3 使用 Zimbra

Zimbra 的 Web 界面相当于功能齐全的邮件客户端，Zimbra 用户群可以通过一个 Web 应用程序访问邮件、联系人信息、日程表和文档。与价格高昂的同类竞争软件相比，Zimbra 优良的品质与性能必然会让众多小规模企业用户感到满意。

■提示：别忘了 Zimbra 有一款免费的桌面客户端，它的功能与 Web 客户端类似。

Zimbra 的 Web 客户端既支持 Ajax 高级特性，又支持标准 HTML 技术。基于 Ajax 的客户端可以展示更为高级的界面呈现效果，但和基于标准 HTML 的客户端相比，前者需要更多的资源。远程用户如果想要获得更好的性能，则可以使用标准 HTML 版本的客户端。使用掌上电脑或移动设备的用户可以使用移动模式（Mobile mode）的客户端，这也是一种可用的 Zimbra 客户端。

■注：想要获得关于 Ajax 的更多内容，请阅读以下站点的文章，见 “[http://en.wikipedia.org/wiki/Ajax_\(programming\)](http://en.wikipedia.org/wiki/Ajax_(programming))”。

我们通过下列 URL 访问 Zimbra 邮件服务器 “<https://mail.example.com>”。图 15-52 显示的是各种可用的客户端模式。Default 模式在 “Creating a Class of Service section of the Preferences” 标签页中设置。在 examplecos CoS 中，设置 Default 模式为 “Standard HTML”（标准 HTML），该模式具有更好的性能。

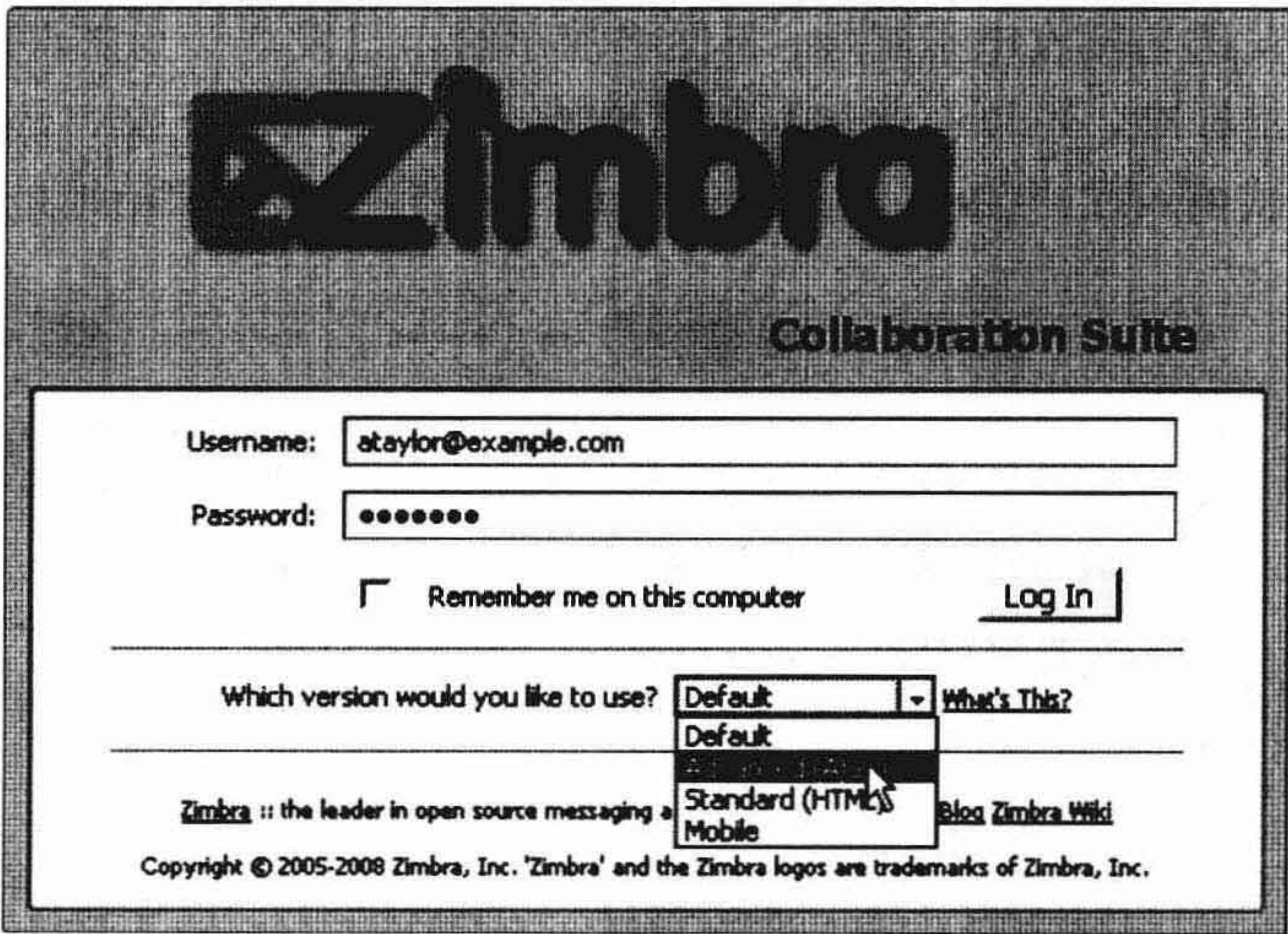


图 15-52 登录 Zimbra Web 客户端

在图 15-52 中，可以看到用户 ataylor 正在使用我们创建的用户名和密码登录，她选择使用 Advanced (Ajax)模式。

15.3.1 使用电子邮箱

选择 Advanced (Ajax)模式，打开用户的 Web 界面，如图 15-53 所示。网页左边是典型的导航面板，主面板顶部是标签导航栏，该导航栏包括 Zimbra 的各个功能和当前正在使用的功能的操作按钮。在标签导航栏上改选别的功能会改变网页左边的面板以及与功能对应的操作按钮。在本例子中，我们示范使用 Mail 功能，读者可以看到这里选择的操作是创建一封新邮件。

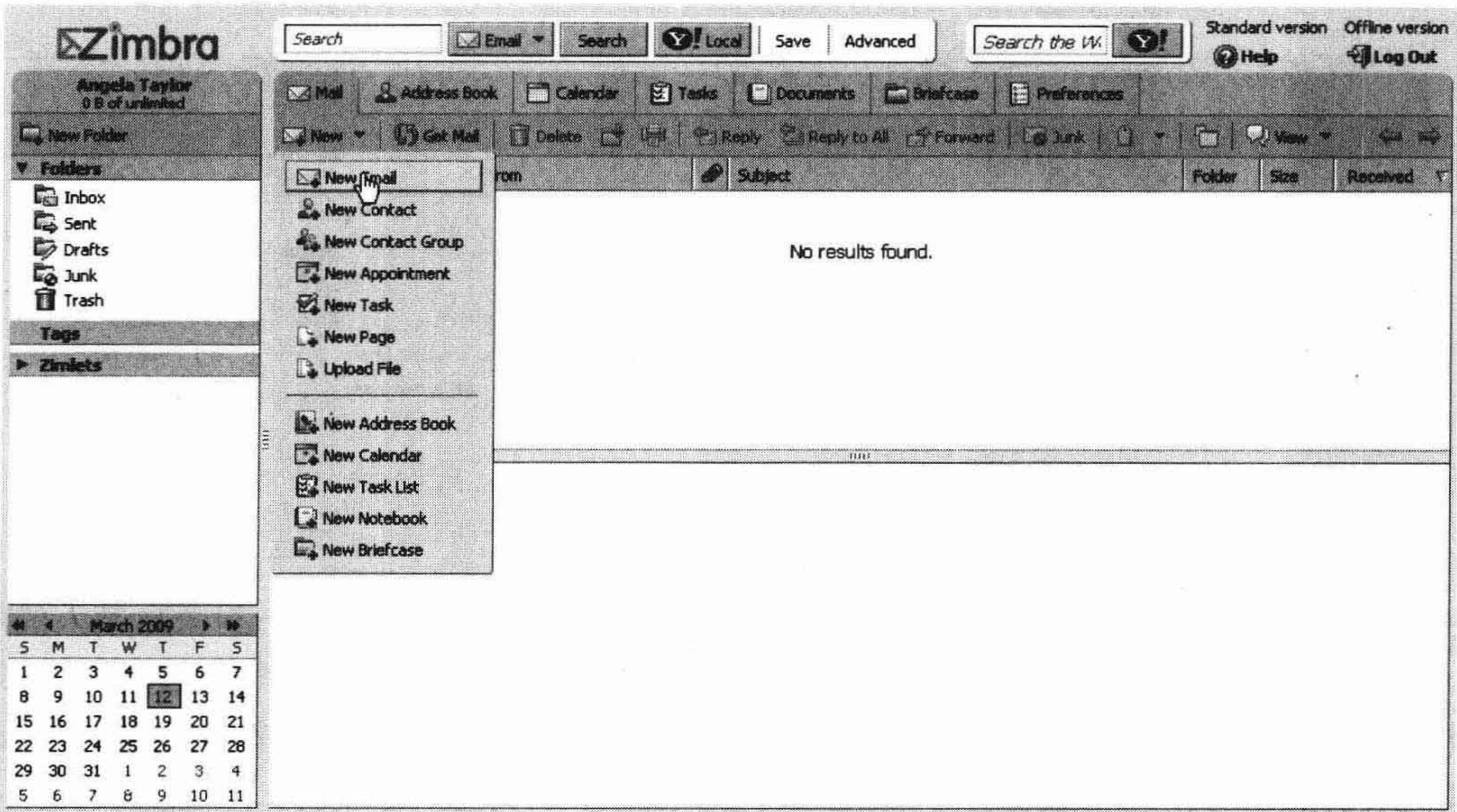


图 15-53 使用 Mail 功能

为了介绍如何发送与接收电子邮件，下面将演示给自己发送一封邮件的过程。如图 15-54 所示，邮箱窗口非常标准。可以在此填写邮箱地址、标题、邮件正文，还可以添加附件。地址字段会自动地尝试填入在“Global Address List”中找到的电子邮箱地址。

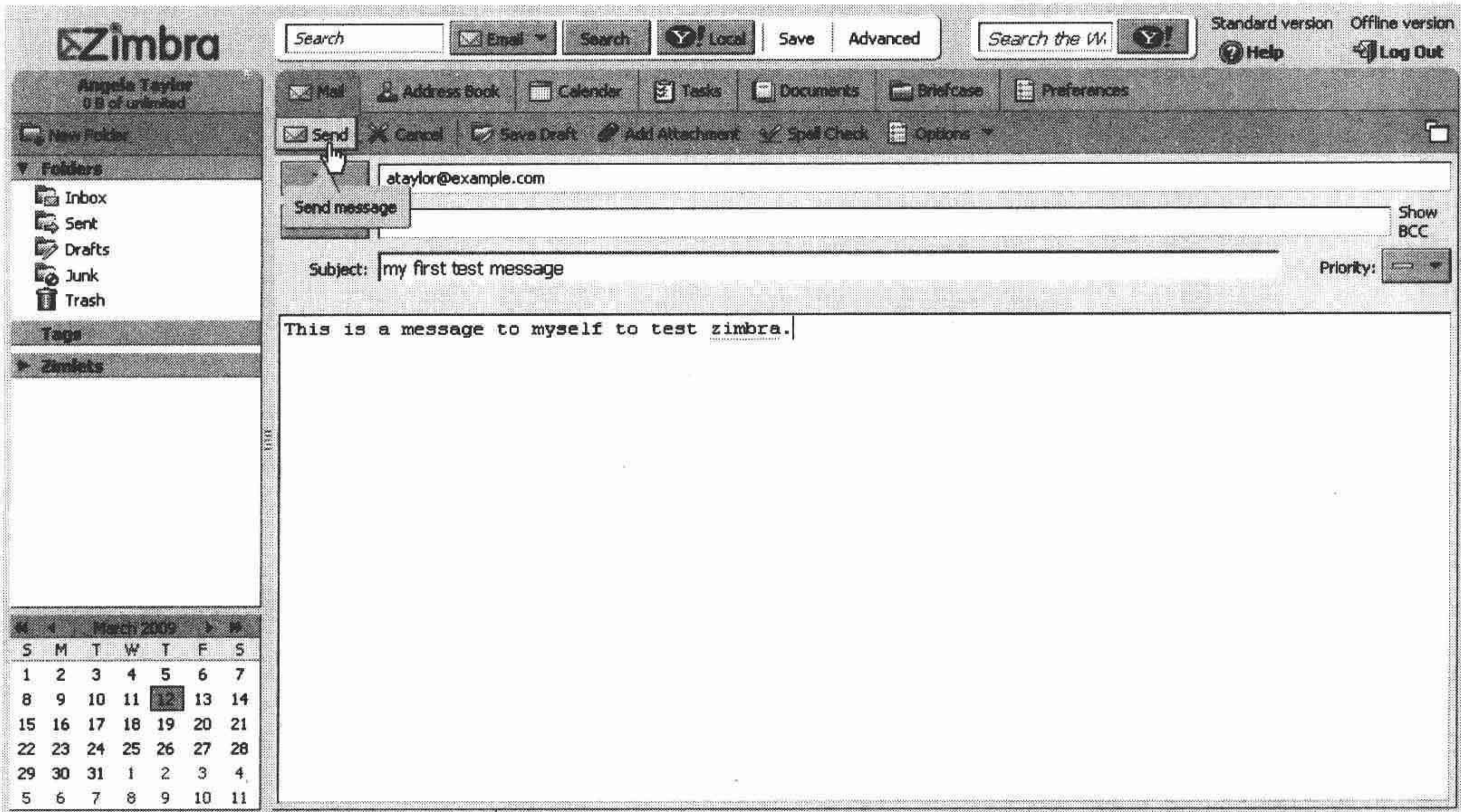


图 15-54 发送一封电子邮件

单击“Send”按钮，这封邮件被发送出去却又被传送回我们的邮箱，如图 15-55 所示，可以看到该邮件被传送回我们的邮箱。邮件窗口显示收件箱列表和邮件预览面板。当双击一封邮件，该邮件的预览面板就会展开。可以使用邮件客户端常用的 Reply（回复邮件）、Reply to all（回复所有邮件）、Forward（转发邮件）、Print（打印邮件）等操作处理邮件。

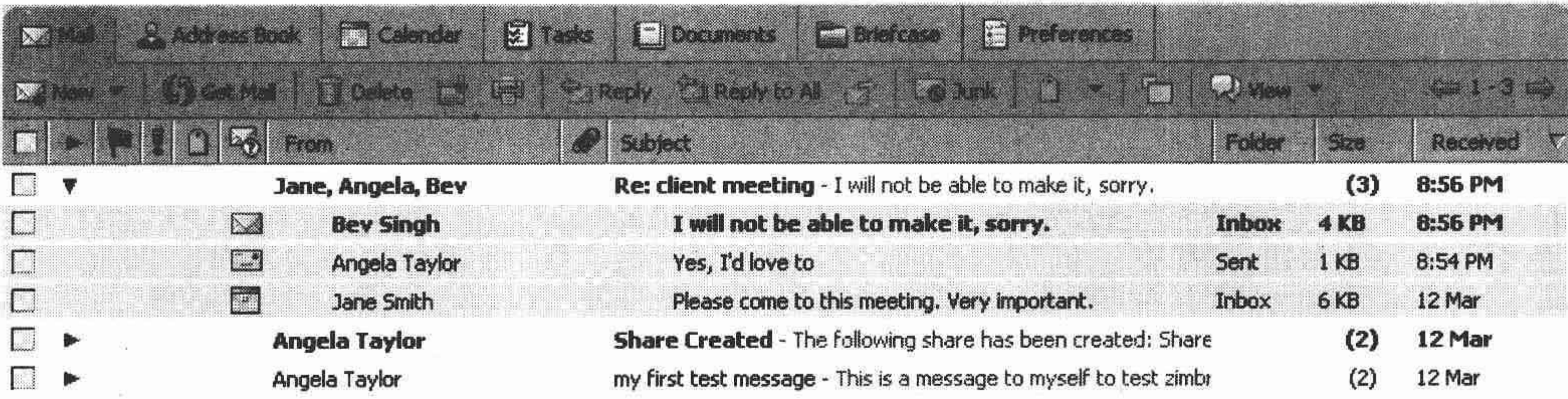


图 15-55 基于会话的邮件组

读者可能并不了解的一个概念是根据会话显示邮件的内容。当收到一封来自某人的邮件时，这封邮件就被当作一次会话的开始。如果回复这封邮件，回复内容就会挂接在初始邮件的下面。其他任何回复内容都会挂接在回复对象的下面，如此等等。读者可以从图 15-55 明白我们所讲的意思。

上图显示参与用户讨论的邮件会话已被展开，如邮件窗口左边那个向下的小箭头的指示。

来自 Jane Smith 的初始邮件是一次会话邀请，Angela Taylor 积极地回复了 Jane Smith 的邮件，而 Bevan Singh 说他不能参与讨论，这就形成一次以那封初始邮件为基础的会话。下一封无关邮件报文发起一次新的会话。

提示：按照会话分组邮件就好比在移动设备上使用文本提示输入功能，有人喜欢，也有人讨厌。当然也可以在 CoS 对象里或者单个用户的首选项设置中设置邮件是否按照会话分组。

另一个读者可能还不熟悉的概念是标签，标签用于把不同的邮件会话快速地组合在一起。标签是一种类似于文件夹的概念，但它可以跨文件夹组合邮件。假设收到许多预约请求邮件，那么可以给所有这样的邮件贴上一个叫做“预约”的标签，然后把这些邮件的实际内容保存到与之相关的业务部门对应的文件夹中。采用这种方法可以简单快速地跨文件夹查看所有贴有“预约”标签的邮件。在图 15-56 中，我们添加一个叫做“my appointments”的标签，用该标签标识那封客户端讨论预约邮件，并把邮件的实际内容保存到 sales 文件夹中。现在，可以单击“my appointments”标签查看所有贴有该标签的预约邮件。

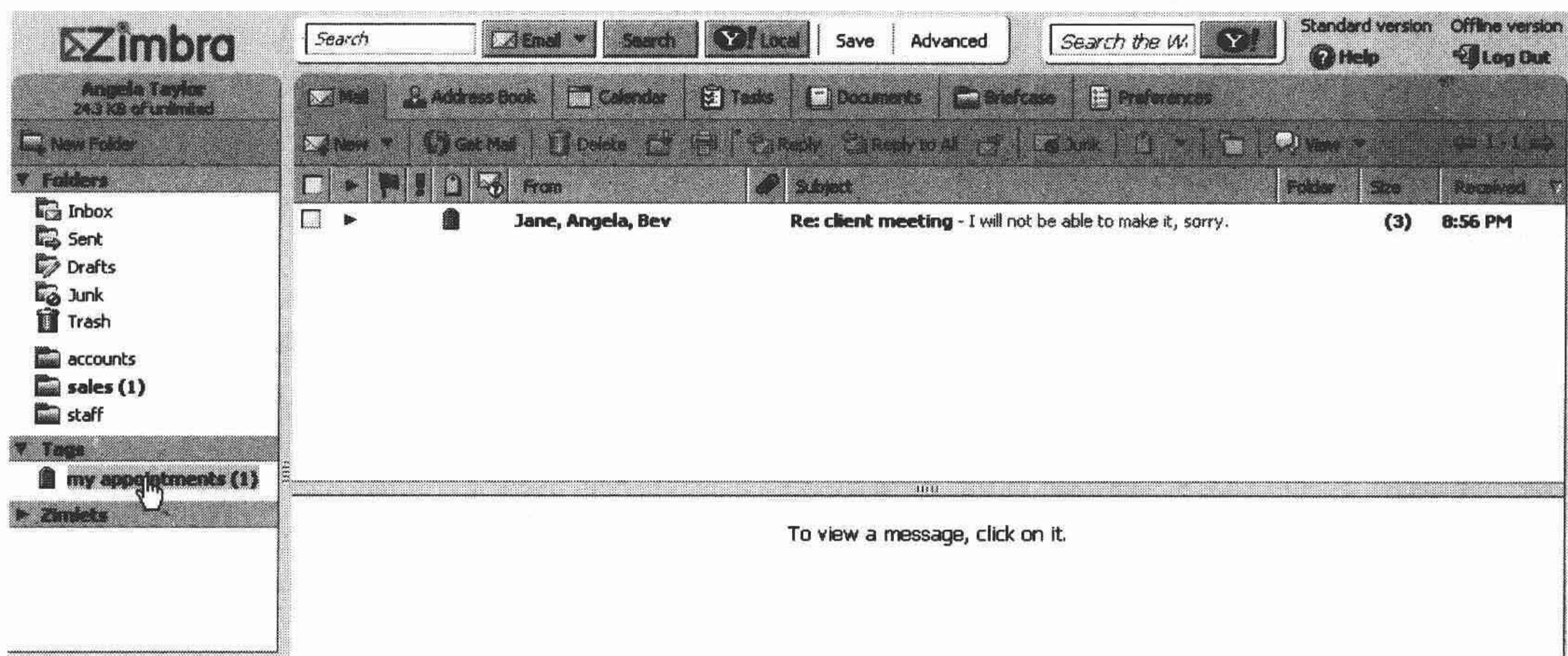


图 15-56 用标签分组邮件

用标签做标识在 Zimbra 的许多功能里都会用到。

15.3.2 使用 Zimlet

之前提到过，Zimlet 为 Zimbra 服务器提供附加功能。Zimlet 通常由 Zimbra 社区成员开发，并和其他 Zimbra 用户共享。现在找到左边导航面板里的 Zimlets 菜单，展开它，即可显示当前所有可用的 Zimlet。图 15-57 显示 Zimbra 服务器上有两个 Zimlet。

名为 Local 的 Zimlet 由 Yahoo!提供——Yahoo!是 Zimbra 新的所有者——它用于显示用户所在区域的本地地图。如果居住在一些边远城市，如澳大利亚的墨尔本，那么就可能会对这个功能感到失望了。对于美国以外的国家，正如读者所看到的，Yahoo! Maps 支持的地图分辨率大概是 30000 英尺。举个例子，把有着 2100 万人口的东京的 Yahoo! 地图和内华达州艾

尔克的 Yahoo!地图做个比较，读者就会明白我们所说的意思。如果用户不在美国，也许会把这个 Zimlet 删掉。

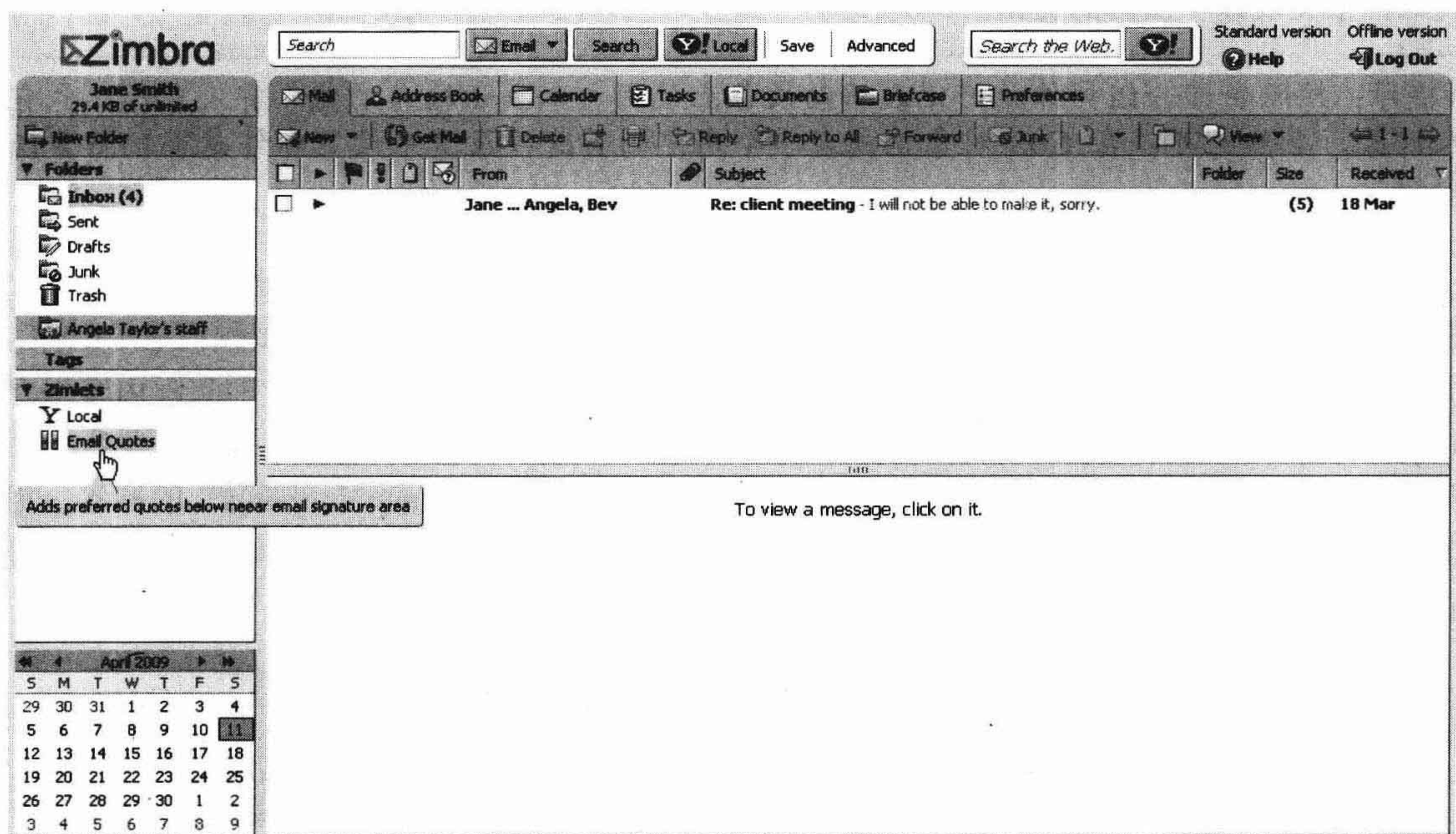


图 15-57 可用的 Zimlet

名为 Email Quotes 的 Zimlet 会根据用户所选的语录类别，在邮件末尾附上一则来自 <http://www.quotedb.com> 的语录。单击这个 Zimlet，会看到一些可设置的选项（如图 15-58 所示）。

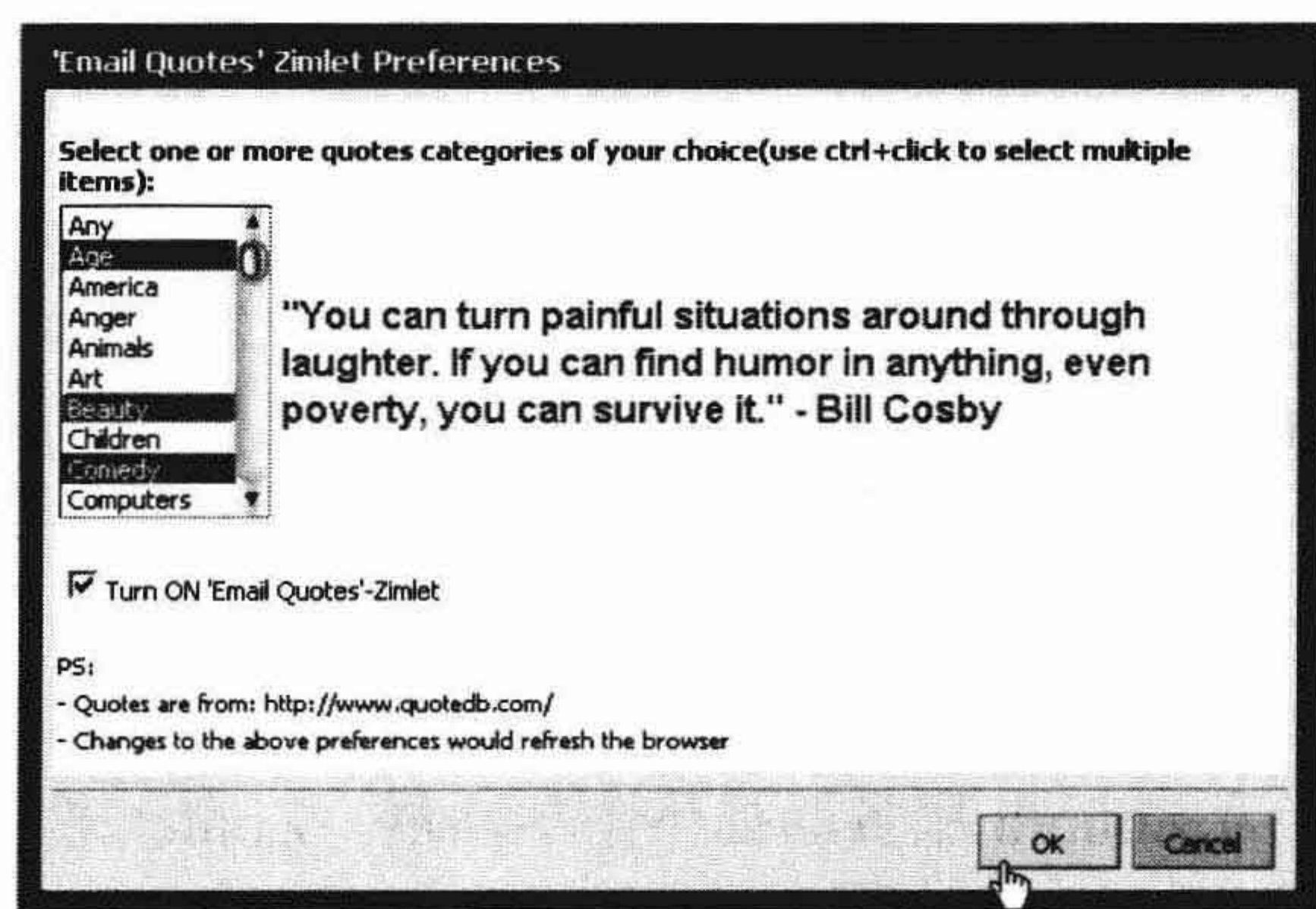


图 15-58 配置 Email Quotes Zimlet

使用鼠标、Shift 或 Ctrl 键，可以选择多个语录类别。通过设置 “Turn ON Email Quotes'-Zimlet” 复选框，还可以开启或者关闭这个 Zimlet。现在，编辑一封新邮件时，该 Zimlet 将向 <http://www.quotedb.com> 发出获取语录的请求。该请求返回一则随机语录，读者将会看到这则语录被追加到邮件的正文，如图 15-59 所示。

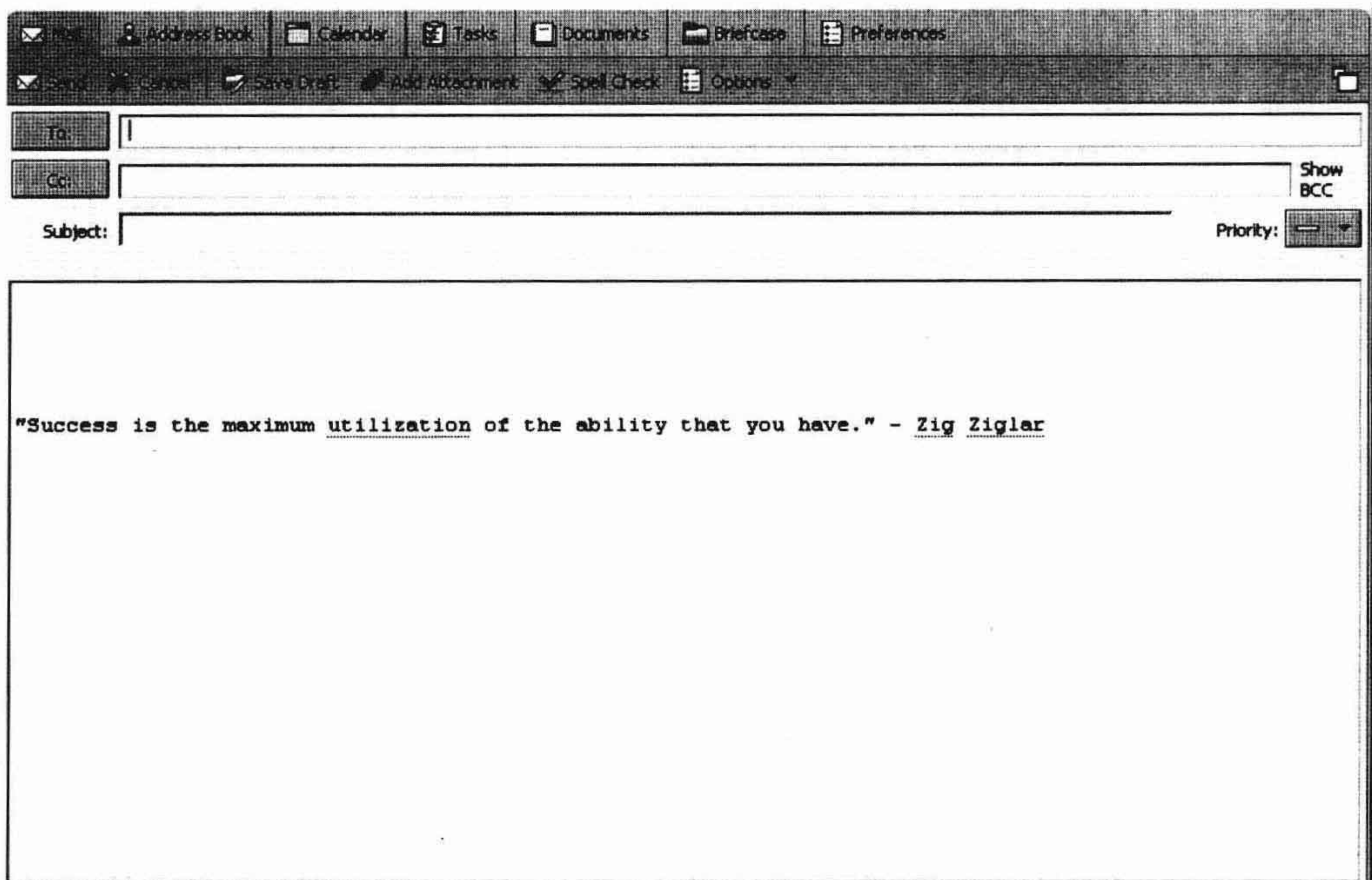


图 15-59 “Email Quotes'-Zimlet”把一则语录追加到邮件的正文

也许这并不是最地道的与商业客户沟通的方式，但是这种方式展示出一个 Zimlet 是什么样的。下面是一些更为有用的 Zimlet。

- Salesforce.com: <http://gallery.zimbra.com/gallery.php?act=viewProd&productId=18>
- Webex Zimlet: <http://gallery.zimbra.com/gallery.php?act=viewProd&productId=51>
- Dimdim Web Meeting: <http://gallery.zimbra.com/gallery.php?act=viewProd&productId=77>
- Asterix PBX Integration Zimlet: <http://gallery.zimbra.com/gallery.php?act=viewProd&productId=94>

此外，建议读者阅读 Zimbra 提供的《Zimlet 导读》，学习该文对 Zimlet 性能的精彩介绍，见“http://www.zimbra.com/demos/zimbra_zimlets.html”。

15.3.3 共享文件夹、地址簿、文档及其他资源

Zimbra 服务器是协作服务器的要素之一，使 Zimbra 用户可以和其他用户共享文件、地址簿、日程表和文档。无论共享对象是什么，在 Zimbra 服务器上共享某项内容的操作都可以按照同样的过程完成。通过 Zimbra，可以轻而易举实现和公司的员工共享邮件夹。比如，Angela Taylor 打算和公司里的其他人共享她的 staff 文件夹，她该怎么做呢？

首先，她使用“New Folder”链接创建一个名叫 staff 的文件夹，“New Folder”链接位于客户端页面左边的“Inbox”菜单之上。为了共享 staff 文件夹，右键单击它并选择“Share Folder”选项。该操作会弹出一个如图 15-60 所示的窗口，叫做“Share Properties”窗口，用户可以在上面指定共享文件夹的一些设置项。

而对于邮件夹，它只能被内部用户或用户组共享。“Role（角色）”规定了共享用户的访问权。在本例子中，我们限制用户只能以 Viewer 角色（这是基本的共享级别）的身份查看共享目录中的邮件。如果希望别人能够在共享文件夹里做进一步的操作，可以赋予他们 Manager 角色或 Admin 角色。

“Share Properties”窗口还允许给别的用户发送标准格式的邮件，邀请他们共享 staff 文件夹，以及从“Message”选项框中选择自定义的邀请邮件。选择与邮件分发列表 allstaff@example.com 共享 staff 目录，意味着公司里的每个职员都可以接收该共享目录并查看其中的邮件。这是一种在公司范围里快速简单地分发邮件的方法，我们只需把收到的任何邮件放进 staff 目录中就可以了。

单击“OK”按钮，邮件分发列表里的每个员工可以选择接受或拒绝该共享目录。在图 15-61 中，注意 Bev Singh，作为 allstaff 邮件分发列表中的一员，他收到共享邀请邮件并准备接受该共享。

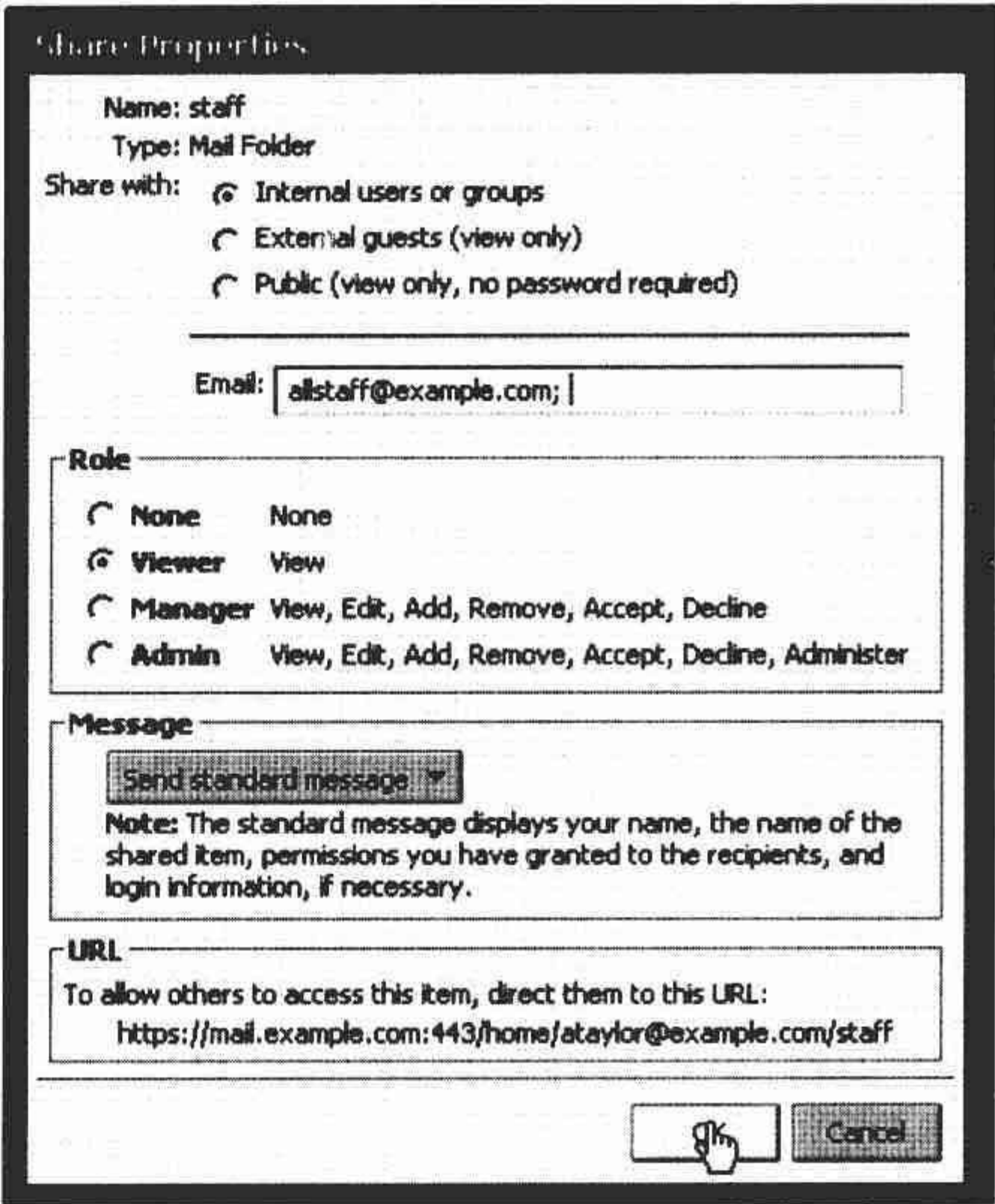


图 15-60 文件夹共享

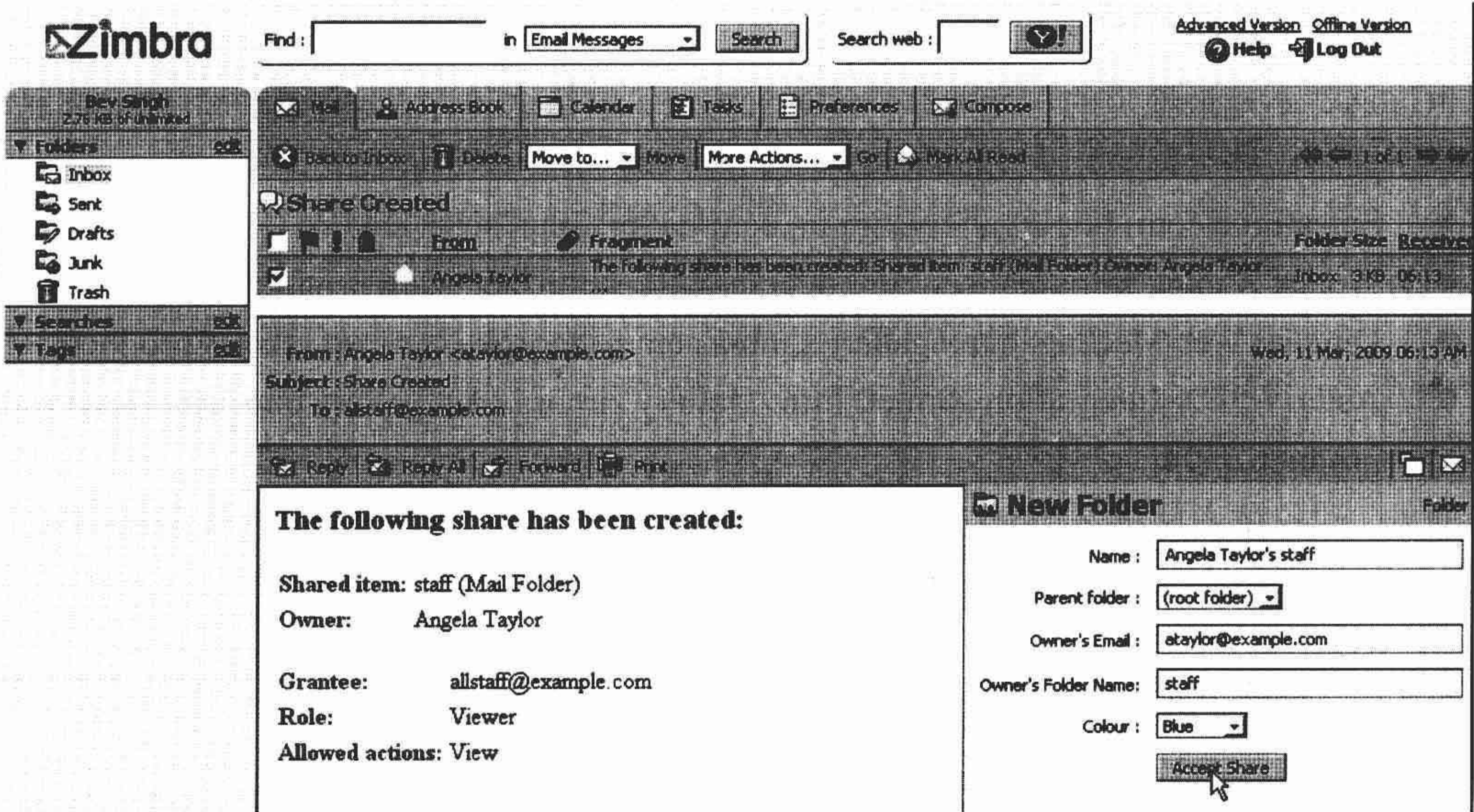


图 15-61 接受共享文件夹

在接受共享邀请之后，共享文件夹就会在 Bev 的收件箱被创建出来。按照同样的步骤，可以共享 Zimbra 服务器上的其他任何内容。除此以外，Zimbra 服务器的共享功能还允许我们进行以下操作。

- 创建一个叫做“主要客户”的地址簿，与其他销售同事共享。
- 在自己的部门范围内共享日程表，以协调会议和活动。
- 设置别人无法查看的私人日程表，或者设置别人只能查看我们何时处于空闲或

忙碌。

- 和自己所在部门或公司其他部门的员工共享文档。
- 允许公司以外的人员查看某些文档。

从这些建议用法中，可以看出我们可以和业务团队的其他成员共享几乎任何东西。

接下来向读者介绍如何创建日程表、共享日程表以及创建预约。首先，打开“Calendar”标签页，单击“New Calendar”按钮，弹出一个对话框，该对话框允许创建一个新的日程表（如图 15-62 所示）。设置对话框上的选项，可以禁止该新建日程表报告我们是否处于空闲或忙碌状态，也可以从远程日程表上同步预约信息以及设置日程表的颜色和名字。可以从图 15-62 中看到，我们准备创建一个叫做“work group”的日程表。

单击“OK”按钮之后，选中新建的日程表，单击鼠标右键，选择“Share”选项，开始该日程表的共享操作过程。在本例子中，我们想和墨尔本的公司同事共享日程表，并允许他们管理该日程表，这意味着他们可以创建预约、接受预约等。图 15-63 显示了如何创建共享日程表。

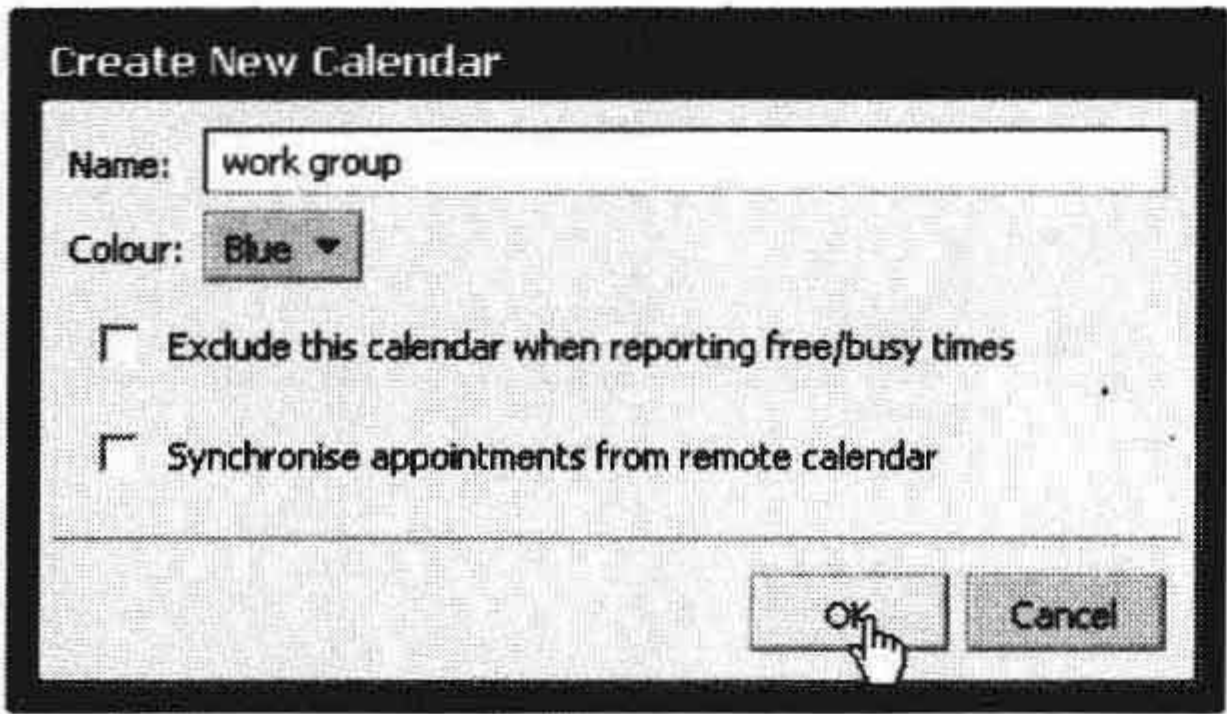


图 15-62 创建一个日程表

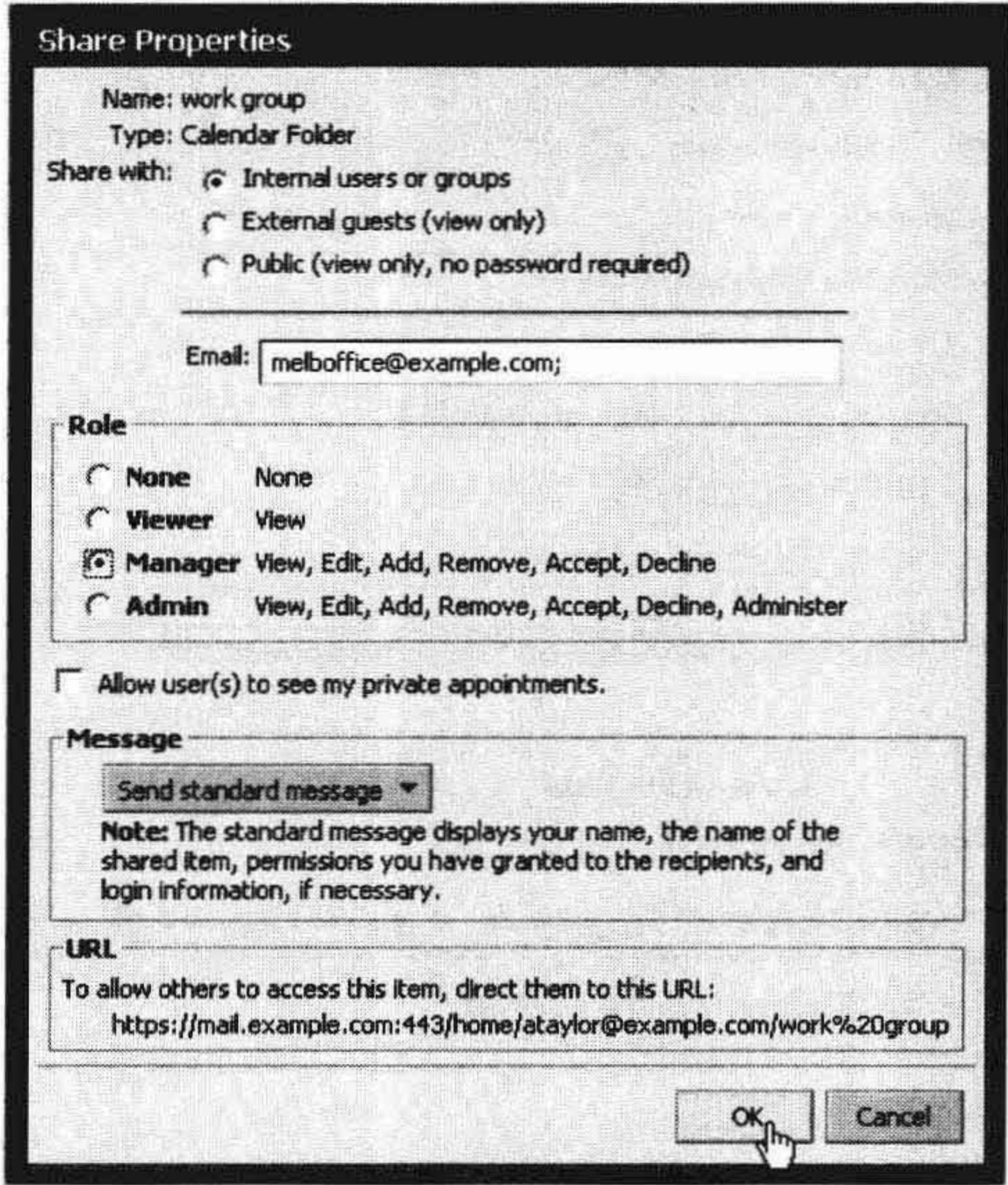


图 15-63 共享日程表

现在，邮件分发列表 melboffice 里的每个人都会收到一封邮件，询问他们是否接受该共享日程表。下面准备创建一个周四上午 9 点的预约，并邀请墨尔本公司里的每个人参加。为此，首先高亮选中一段合适的时间，如图 15-64 所示。

接着，弹出“QuickAdd Appointment（快速添加预约）”窗口。单击“More detail（更多设置）”按钮，为举办会议进行一些设置，如图 15-65 所示。

读者可以看到，这里创建的预约标题为“meeting about calendar”，预定 boardroom 资源作为会议室，邀请 melboffice 邮件分发列表里的每个人参加会议，请务必选择 work group 类型的“Calendar”。然后保存该预约，如图 15-66 所示，该预约将显示在日程表的彩色时间段里。

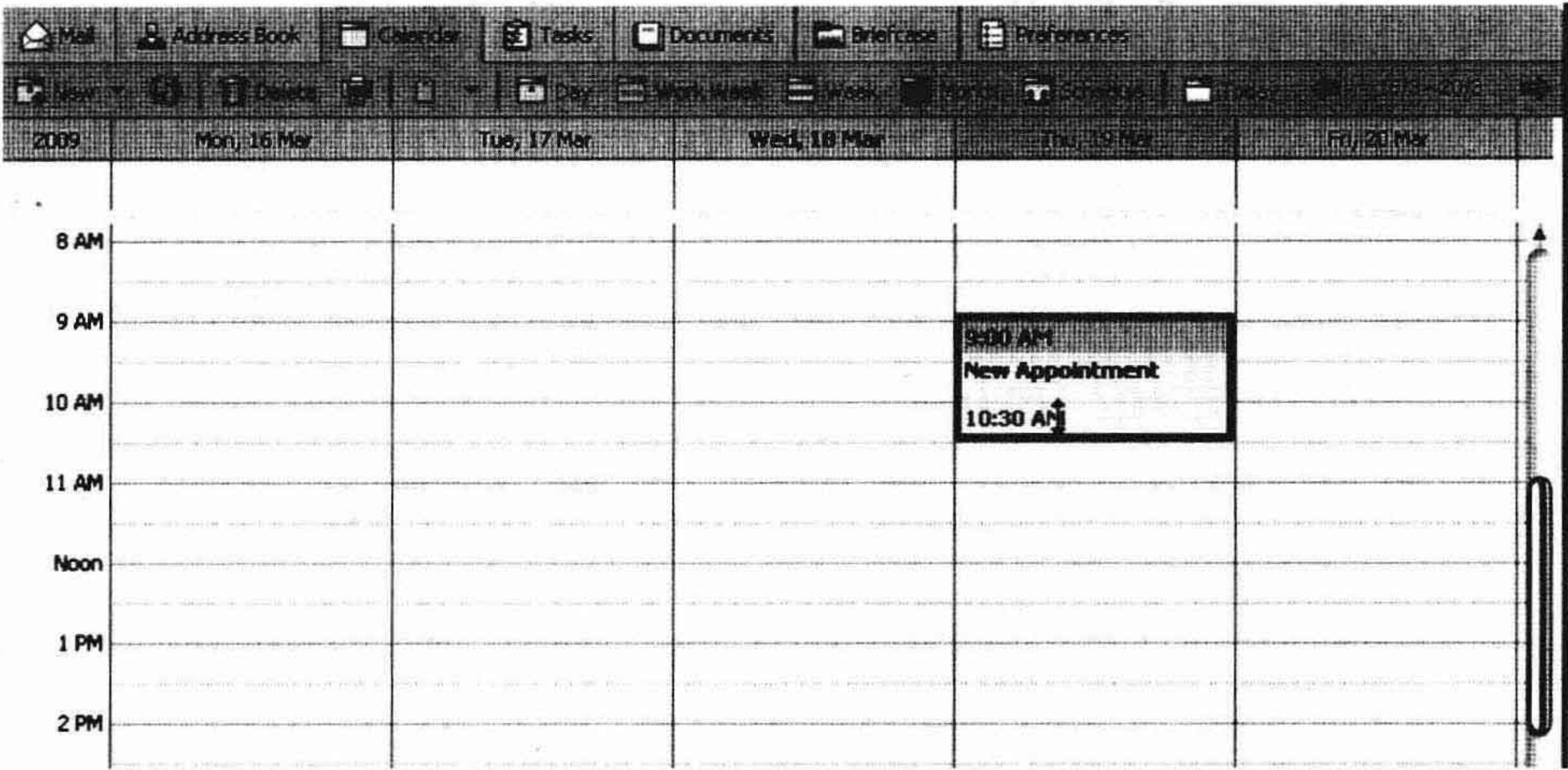


图 15-64 标记时间段

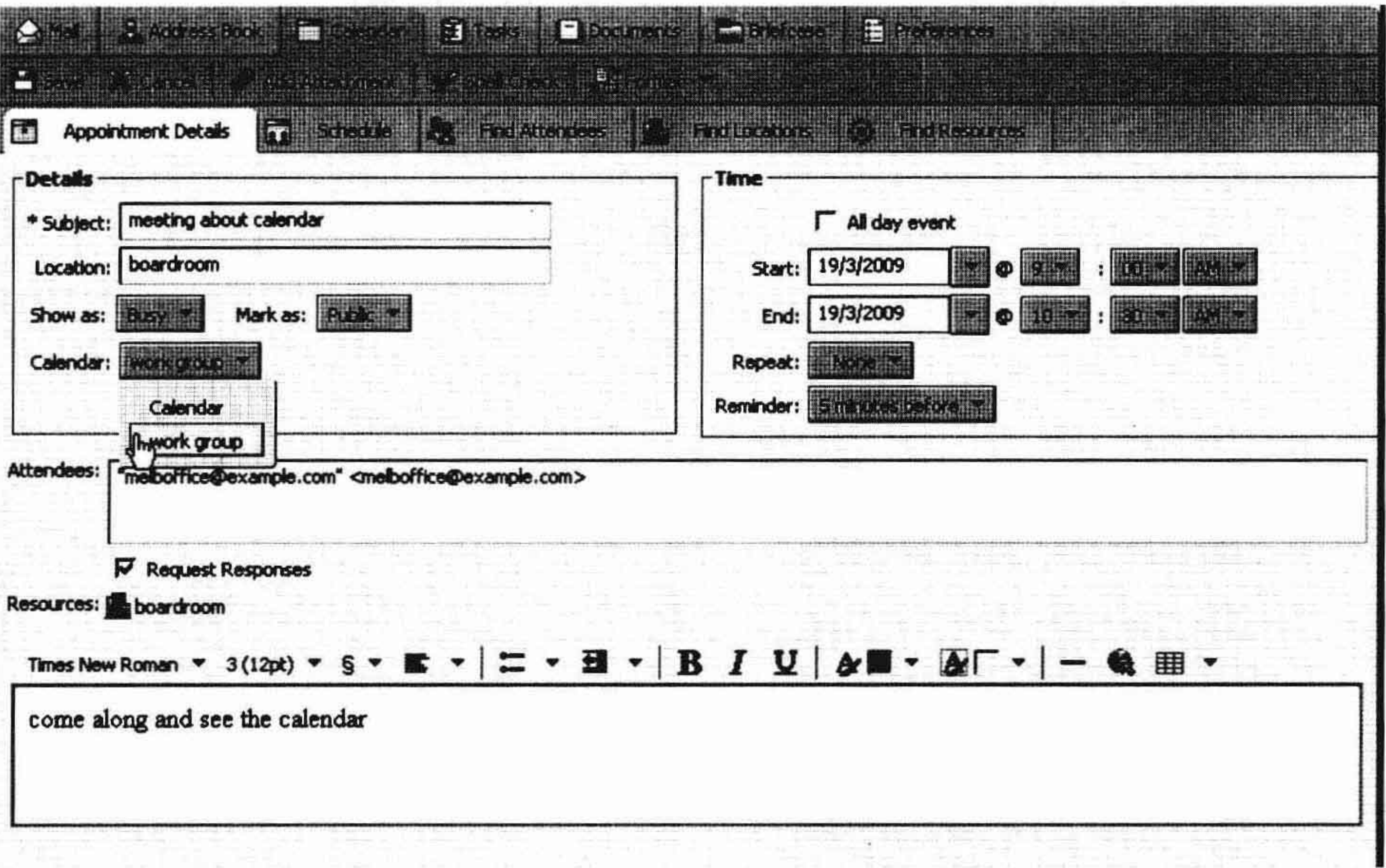


图 15-65 创建预约

现在，可以移动、延长或者删除该预约时间段，并且该预约的镜像将显示在每个人的日程表里。

紧挨着“Calendar”标签页的是“Tasks”标签页。在该标签页上，可以创建和共享任务，如个人任务列表或共享任务列表。该标签页具有基本的工作流功能和时间管理功能，这些功能可用于跟踪已分配任务的进度。和 Zimbra 服务器上共享文件夹和日程表的过程一样，我们可以按照此过程共享这些任务。

现在向读者介绍“Documents”标签页。可以在该标签页上创建文档，并可以在公司范围内或者在一个广泛的类 wiki 系统环境里共

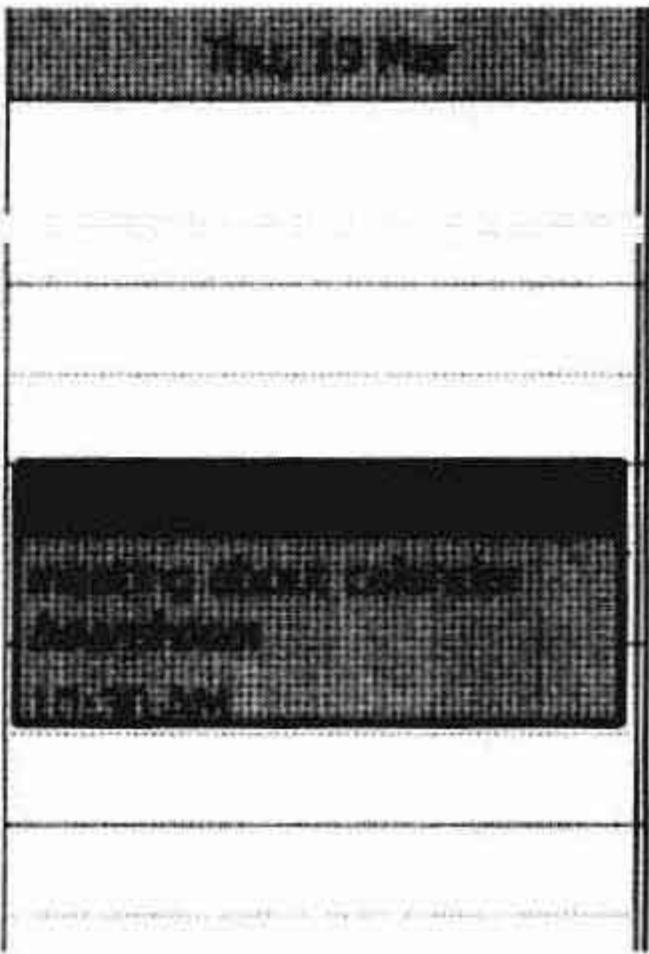


图 15-66 已创建的预约

享所创建的文档。

转到“Documents”标签页，如图 15-67 所示。

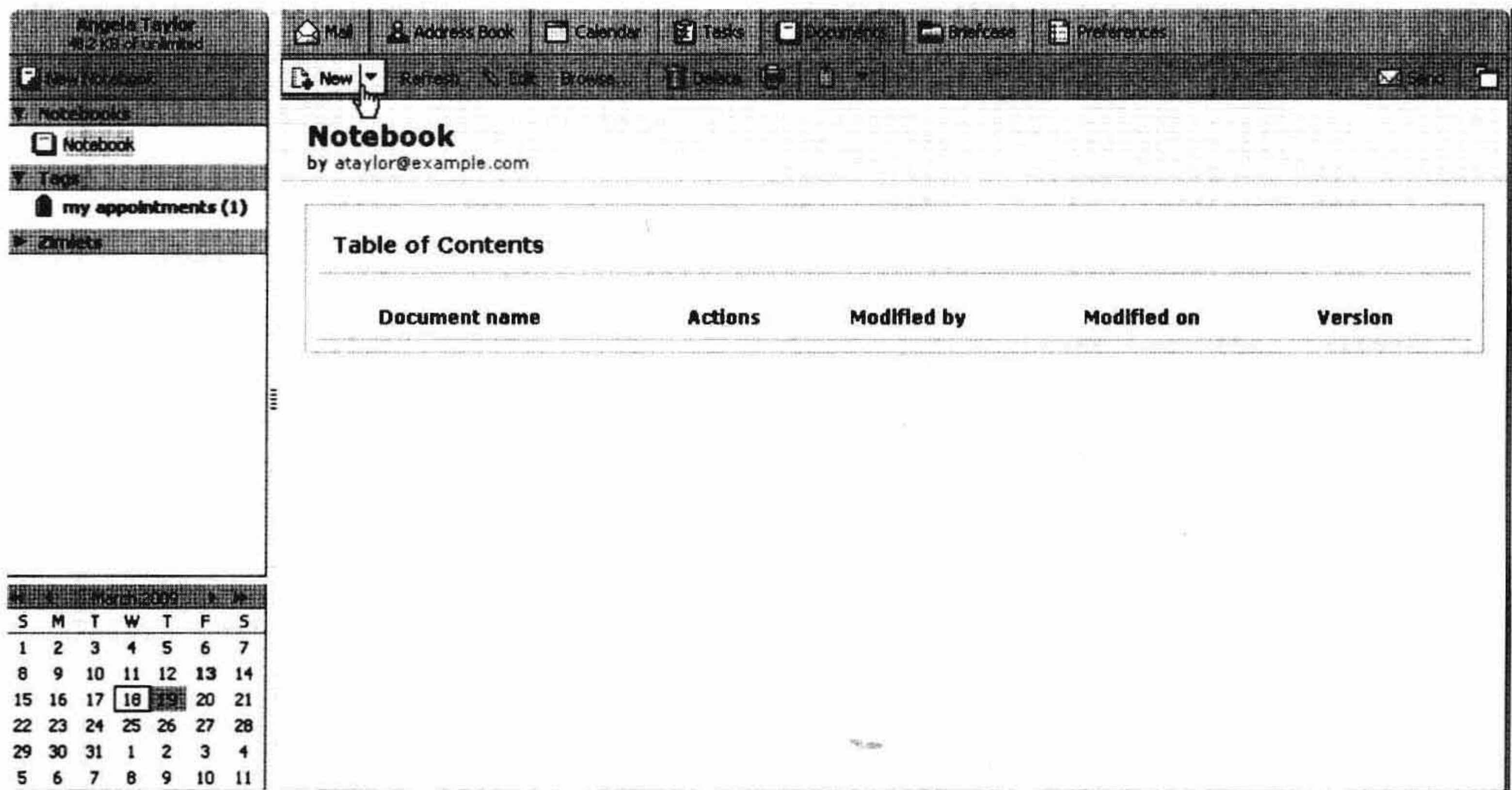


图 15-67 使用文档

正如读者在左侧面板所看到的，可以把工作分组放进“Notebooks”栏。这里所创建的每一个记事本都可以被公司里的职员共享，或者如有需要还可以被不属于公司的外部人员共享，这个文档共享工具的作用在于存放在记事本里的文档可以包括公司的客户问答、用户文档，或者是用户可以利用的其他有用信息。

接下来向读者介绍如何在已有的记事本上创建新的记事页。选中“New”按钮，弹出一个编辑窗口，使用该窗口创建一个新的文档。作为演示的目的，本例创建一个快速 FAQ，如图 15-68 所示。

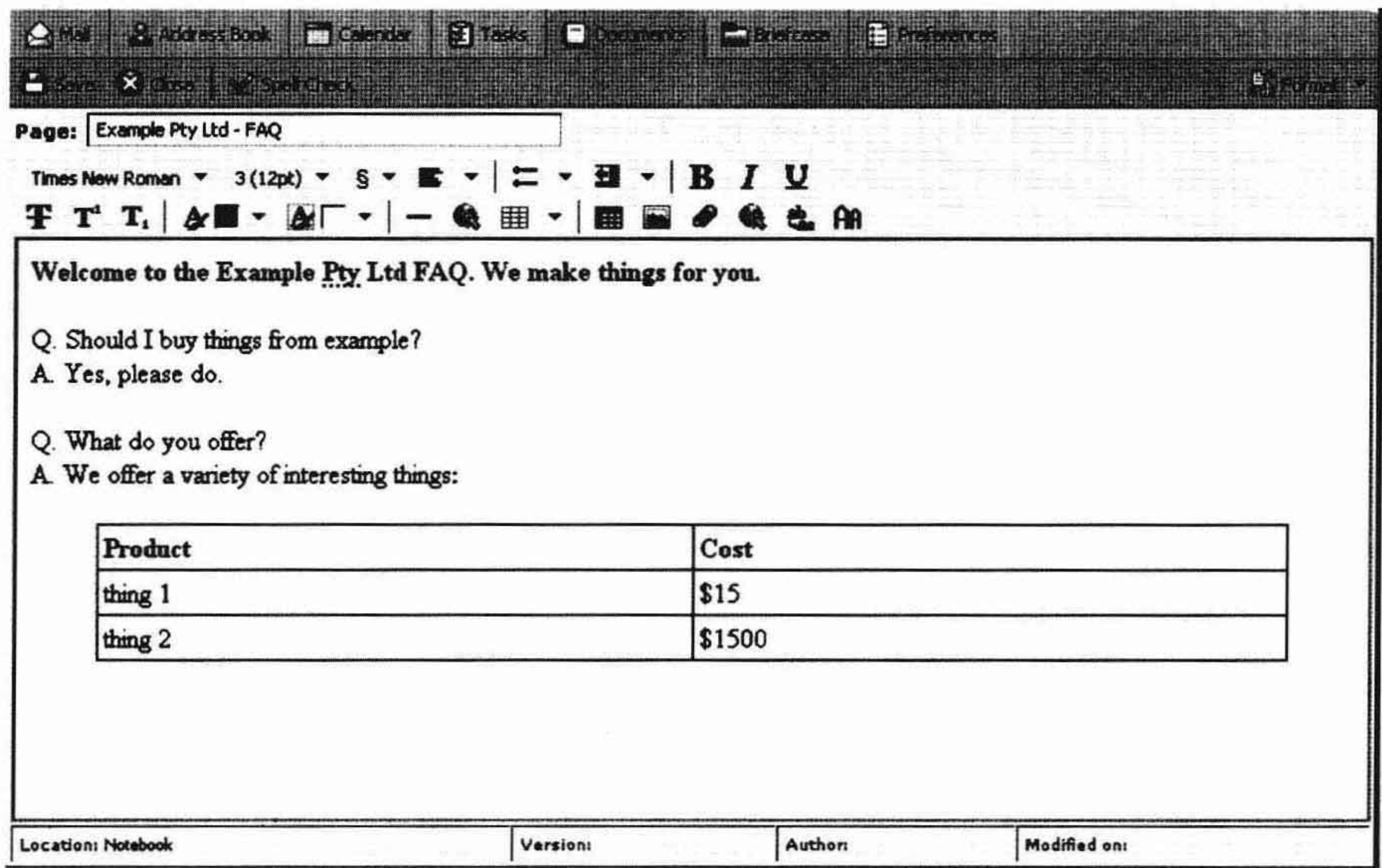


图 15-68 创建一个快速 FAQ 文档

正如读者所看到的，很容易在 Zimbra 上创建一个文档，和许多 wiki 不同，我们不必学习其他的编辑语法，就可以非常简单快捷地使用规范样式创建表单和电子表格。当编辑完成时，保存该文档，就可以返回该记事本的 TOC（Table of Contents，内容列表），如图 15-69 所示。

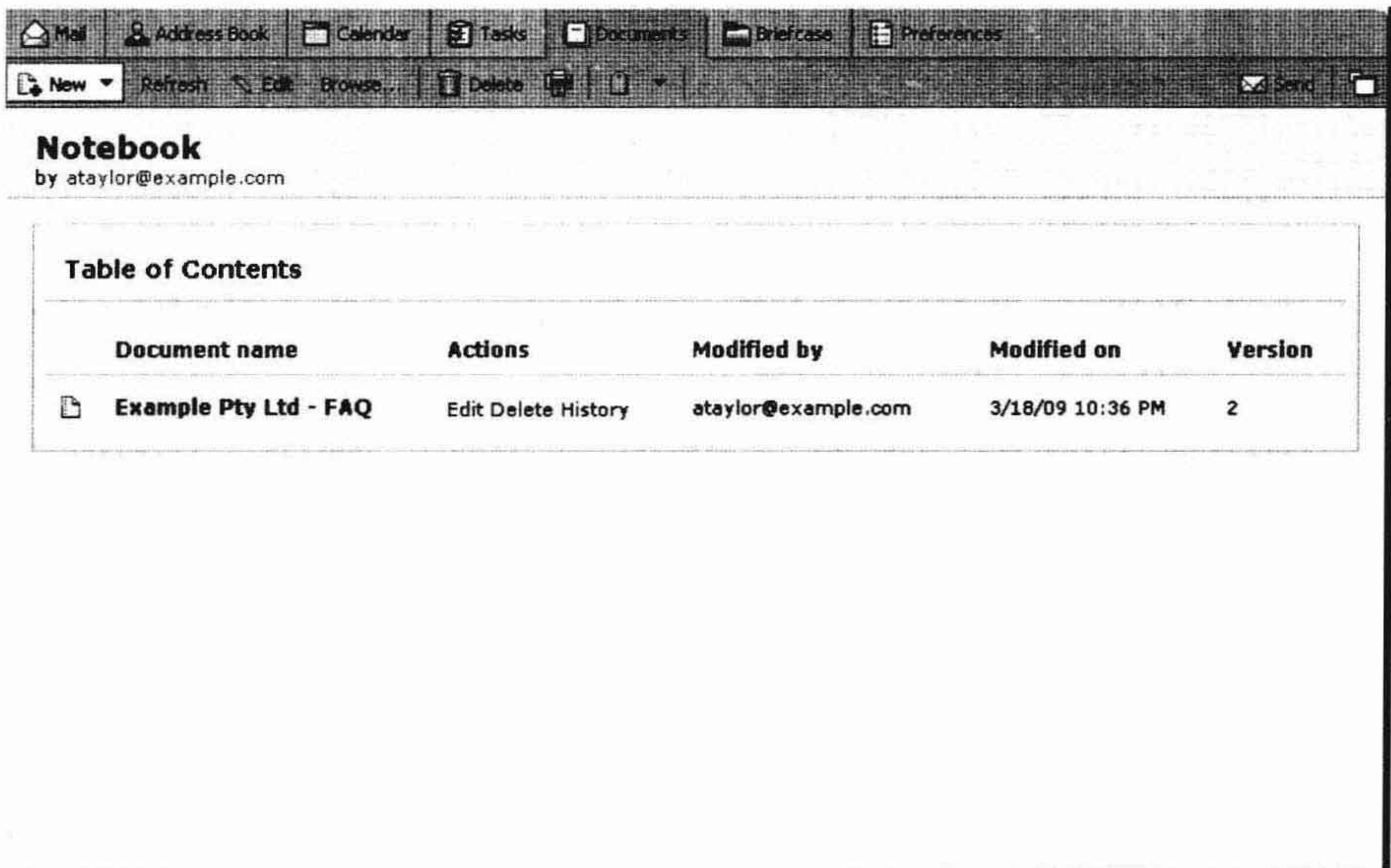


图 15-69 记事本的 TOC

现在已有一个记事本以及一个描述记事本文档内容的 TOC。如果查看所创建的记事页的信息，将会看到这些信息包括版本信息和修改信息。我们可以把记事本功能用于公司内部或外部文档，鼠标右键单击前面创建的记事本，即可弹出可用的共享选项框。

最后向读者介绍的是“Briefcase（公文包）”功能。Briefcase 用于保存电脑上的文档，这样无论在任何地方，都可以轻松获取它们。图 15-70 显示的是将要共享的一个 Briefcase，在这个 Briefcase 中有一个 Word 文档。

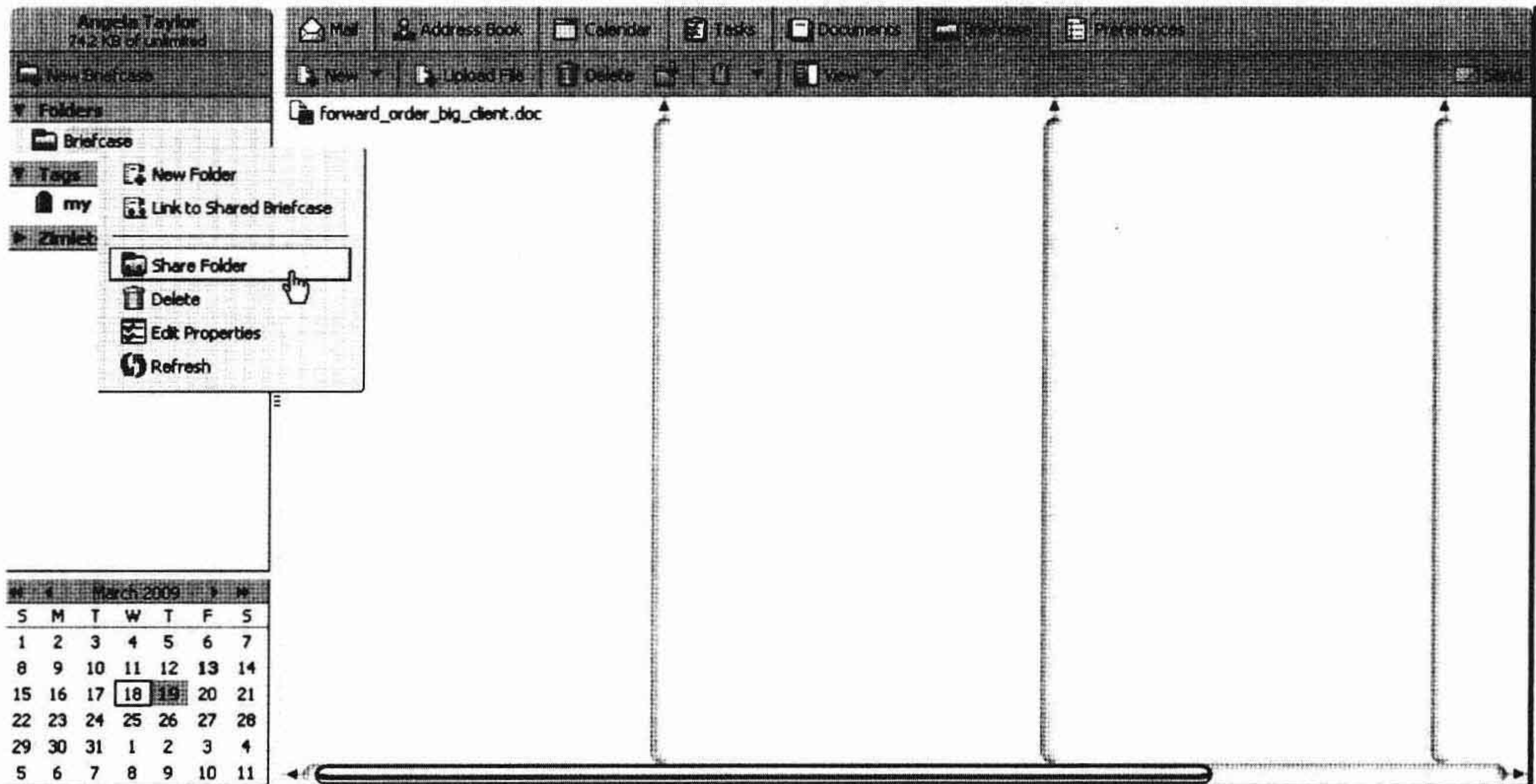


图 15-70 我们的 Briefcase

根据需要，可以创建任意多个 Briefcase，还可以和公司里的任何人共享它们。Briefcase 有一个很棒的功能，即单击 Briefcase 中的某个文档，可以通过邮件发送该文档的一个链接。

15.4 迁移已有的邮件服务

有时候会遇到这种情况，公司已有一个邮件系统，而我们需要把用户的现有邮件迁移到新的 Zimbra 服务器上。这里，迁移的方法取决于所要迁移的邮件系统。Zimbra 提供两种迁移工具，它们可以从管理控制台下载，读者也可以在图 15-71 中看到这两种工具的下载界面。

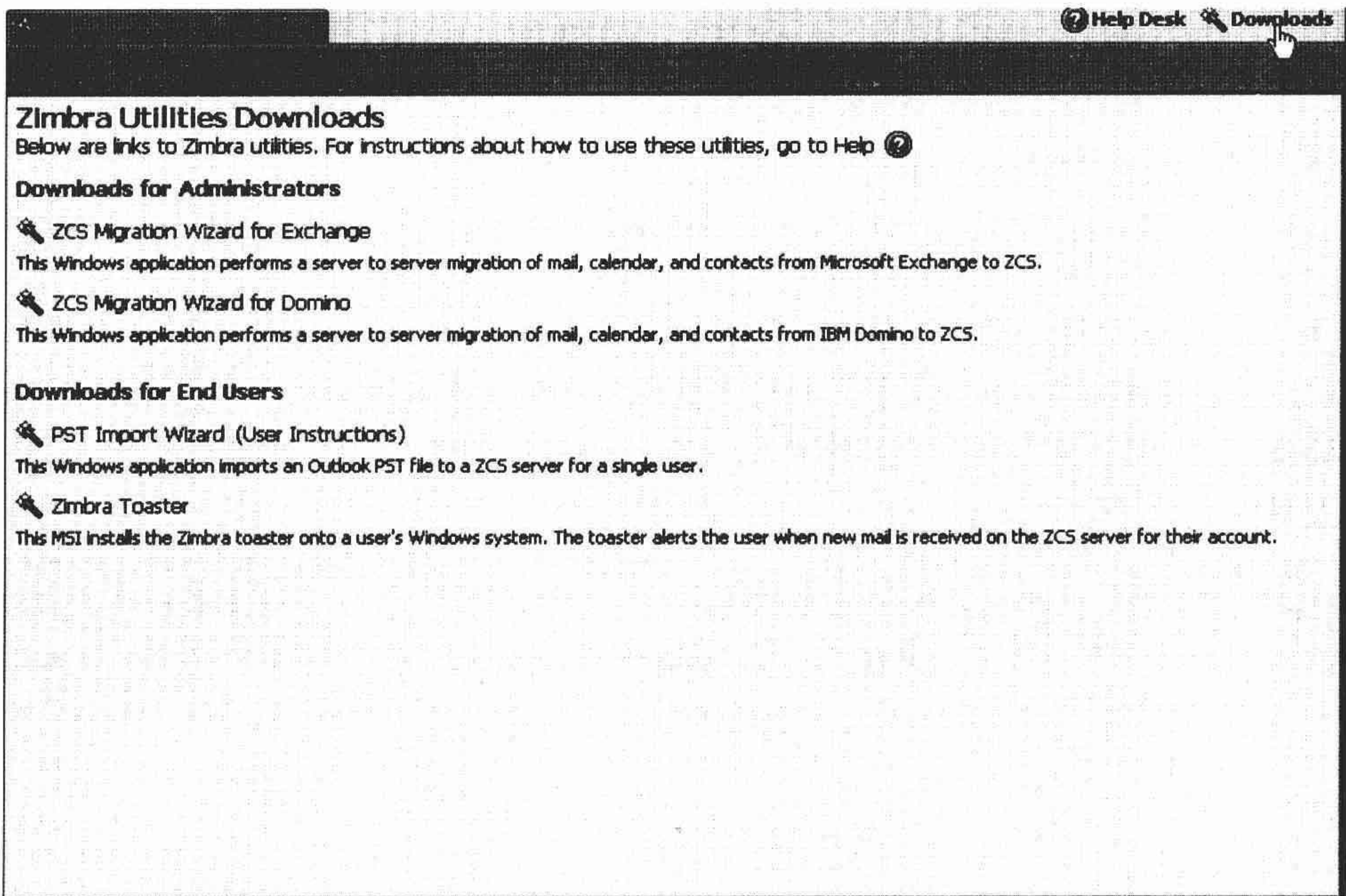


图 15-71 迁移工具

在管理控制台上，可以在右上角看到一个 Downloads 链接，该链接提供迁移工具的下载。前两个下载项中，一个是 Microsoft Exchange 服务器的下载，另一个是 IBM Domino 服务器的下载。后两个下载项用于终端用户，一个用于把 PST 文件（保存在邮件服务器上的 Microsoft Outlook 文件）迁移到 Zimbra 服务器，另一个用于当有新邮件到达终端用户的 Zimbra 账号时发出提醒。

Zimbra 在以下网址提供把数据从 Microsoft Exchange 服务器迁移到 Zimbra 服务器的文档：http://www.zimbra.com/docs/os/latest/migration_wizard_for_exchange_installation_guide/。

那些想把备用的 IMAP 服务器迁移到 Zimbra 服务器的用户可以使用 imapsync 工具。Zimbra wiki 在以下网址提供关于如何完成此种迁移的详细文档：http://wiki.zimbra.com/index.php?title=User_Migration。

imapsync 工具允许我们在绝大部分的 IMAP 服务器之间迁移邮件。方法是使用 imapsync 连接一个 IMAP 服务器，然后把某个账号的邮件转移到 Zimbra 服务器的某个账号。所以用户

必须在 Zimbra 服务器上创建好相应账号。可以在这里下载 imapsync 工具：[http:// freshmeat.net/projects/imapsync/](http://freshmeat.net/projects/imapsync/)。

Ubuntu 系统的在线资源库提供了 imapsync 软件包，所以可以输入 aptitude 命令安装该工具。

```
$ sudo aptitude install imapsync
```

Red Hat RHEL 主机要求我们从 Freshmeat 网站上下载 imapsync 软件包，并安装所有该软件包必需的依赖软件（请看 INSTALL 文件中的依赖软件列表）。CentOS 系统和 Fedora 系统可通过网站 RPMForge (<https://rpmforge.net>) 的 Yum 资源库提供 imapsync。

■提示：从另一个邮件服务器上迁移邮件，可能需要增加邮件附件大小的默认值。可以通过用户 zimbra 执行下列命令来增大该数值。

```
$ zmprov mcf zimbraMtaMaxMessageSize 50000000
```

可以按照下列命令使用 imapsync。

```
$ imapsync --host1 orig.imap.host --user1 username --passfile1 /etc/password ➡  
--host2 mail.example.com --user2 username --passfile2 /etc/password
```

使用 imapsync 命令，指定迁移邮件的源 IMAP 主机，提供该主机某个用户的用户名和密码。然后，指定迁移目的 Zimbra 主机的相关信息，并提供该主机的用户名和密码。--passfile1 和 --passfile2 选项用于指定包含用户密码的文件，通过把该文件的权限设置为 600，可以把它安全地存储在主机上。

■提示：如果有多个用户的大量邮件需要迁移，则可以采用批量迁移。请查看关于用户迁移 (User Migration) 的 wiki 网页，获得更多信息，见 “http://wiki.zimbra.com/index.php?title=Main_Page”。

15.5 小 结

到目前为止，读者差不多已了解使用诸如 Zimbra 之类的协作服务的种种好处。Zimbra 并不是市场上唯一的协作服务器，但我们认为它是一款易于使用、功能完备的协作服务器。

本章向读者介绍了如何进行下列操作。

- 安装和配置 Zimbra 服务器。
- 添加用户账号、邮件分发列表和资源到 Zimbra 服务器。
- 使用 Zimbra Web 客户端发送邮件。
- 使用 Zimbra Web 客户端共享文件夹、日程表、文档和地址簿。
- 创建日程表和日程表活动，并邀请别人参加。
- 创建文档。
- 使用 Briefcase（公文包）功能。
- 把用户账号从一个电子邮件系统迁移到 Zimbra。

下一章向读者介绍 Active Directory 的开源替代产品：OpenLDAP，并讲解目录服务。

第 16 章 目录服务



目录服务在主流的计算机网络上被广泛地应用着。LDAP 目录就是这种服务的一个例子。LDAP 目录是含有用户名、密码、通用名 (common name)、e-mail 地址、办公地址以及其他属性的特殊的数据库。企业最初使用目录服务是为了便于地址簿和用户信息的分发。由那以后, 目录服务慢慢承担起了企业用户信息和认证服务中心仓库的角色。目前已经开发出了具有针对目录服务进行认证功能的应用程序, 从而更加提高了目录服务在企业中的重要性。

本章将介绍如何安装和配置一个 OpenLDAP 服务器, 还将讨论如何通过添加自定义的模式来扩展 OpenLDAP, 同时还会展示如何设计访问控制列表来增强所安装目录服务的安全性, 以及如何通过命令行和基于 Web 的 GUI 来配置 LDAP 服务器, 其中包括部署单一登录(single sign-on) 服务、Apache 网络认证服务以及基于网络的应用程序认证服务。

目录服务的部署可能会是比较复杂的。尽管其安装非常简单, 但完成安全的配置却通常是非常复杂的。OpenLDAP 没有商业支持的版本, 但是即使是在 OpenLDAP 的邮件发送清单中提出的最简单的问题都会有资深的项目工程师和开发者给出完整的回答 (邮件发送清单是一个巨大的帮助资源, 也是对他们所作贡献的绝对赞誉)。尽管如此, 在开始安装并深入理解软件之前, 购买一本专注于项目的参考书将得到更好的服务。推荐参考以下书籍。

《部署 OpenLDAP》, Tom Jackiewicz 著 (Apress 出版社, 2004 年出版)

《精通 OpenLDAP: 配置, 安全和整合目录服务》, Matt Butcher 著 (Packt Publishing 出版, 2007 年出版)

■注: OpenLDAP 在 “<http://www.openldap.org>” 的网站也提供了一个非常好的管理员指南和常见问题解答 (FAQ)。

16.1 什么是 LDAP

LDAP, 表示轻量级目录访问协议 (Lightweight Directory Access Protocol), 用于访问源于目录访问协议 (DAP) 的基于 X.500 的目录服务。X.500 是一系列协议, 这些协议概述了用户信息的存储和访问方式。LDAP 来源于无 TCP/IP 支持能力的目录访问协议 (DAP)。

■注:要了解关于 X.500 OSI 协议的更多信息,请访问网站“<http://en.wikipedia.org/wiki/X.500>”。

有几种常见的目录服务,它们都是源于 X.500 DAP OSI 模型,例如微软的活动目录 (Active Directory)、Red Hat 的目录服务以及 Sun 的企业版目录服务。本章中,集中讨论常用的、非常健壮的 OpenLDAP 服务。OpenLDAP 是最初由密歇根大学设计的一个原创项目的分支,目前由 OpenLDAP 项目组的工程师和开发者组成的团体所维护 (项目网站为 “<http://www.openldap.org/project/>”)。

X.500 目录访问协议的 OSI 模型阐述了一些 LDAP 所遵循的基本概念。首先,需要有一个独立的目录信息树 (DIT)。这是一个层次化的节点结构,每一个节点需要有一个唯一名称 (DN) 标识。这个唯一名称 (DN) 标识由相对唯一名称 (RDN) 标识和父节点标识组成。DIT、DN 和 RDN 之间的基本关系如图 16-1 所示。

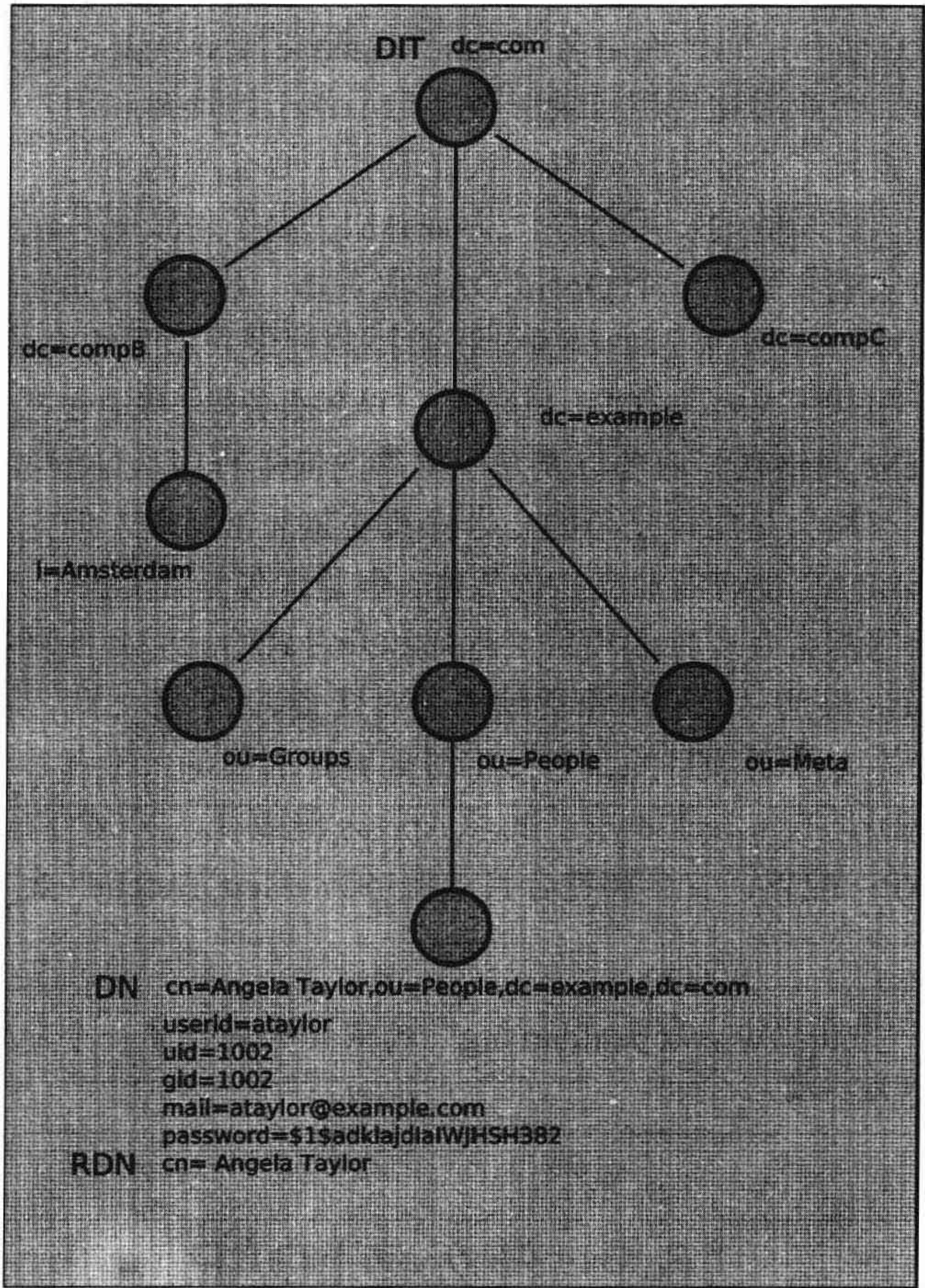


图 16-1 DIT、DN 和 RDN

DIT 是目录树,在这里它有一个名为 `dc = com` 的根 DN。有几种方式可以定义根和主分支,有人选择基于它们 DNS 域名的布局,就像这里给出的一样;有人使用地理位置作为根,如 `o = US`、`o = AU` 或 `o = DE`。在示例中,因为没有过度地考虑公司的地理位置,选择使用

DNS 域名命名标准。如果愿意，也可以在树的下层引入位置名称属性，就像在示例中的 `compB` 分支中指定了一个“`l = Amsterdam`”位置一样。应该善于对如何布局目录结构进行很多考虑，但是最终目的是尽可能地使目录结构易于理解。

■提示：现在就定义一个命名分支和描述公司结构的标准方式，并且要一直坚持使用这种标准方式，这一点很重要。

分支用来将信息组织成逻辑组。这些逻辑的组称为组织单元（organizational unit）并表示为 `ou`。可以把喜欢的任何内容组织在一起，然而在 LDAP 树中通常能看到的主要的组织单元是人员（People）、组（Groups）和机器（Machines）。把和人员有关的内容保存在“`ou = People`”分支下，用户组保存在“`ou = Groups`”分支下，非人力设施保存在“`ou = Machines`”（通常也命名为 `ou = Hosts`）分支下。尽管建议在设计自己的 DIT 时越简单越好，但是在逻辑上一个组织单元可以包含另外一个组织单元，而且同时想让它们多复杂它们就可以有多复杂。

Root 下的每一个记录的 DN 都是唯一的，DN 由 RDN 及其父节点名组成。在图 16-1 中可以看到，由一个 DN 为“`cn = Angela Taylor, ou = People, dc = example, dc = com`”的节点，它由名为“`cn = Angela Taylor`”的 RDN 和父节点“`ou = People`”、“`dc = example`”以及“`dc = com`”组成。同样的，“`ou = People, dc = example, dc = com`”也可以作为人力组织单元的 DN，其 RDN 为“`ou = People`”。

每一个 DN 节点都由类名和描述此节点的属性构成。对象类规定了对象应该具有或允许具有何种属性。类还有可能支持其他的类，以便于给它们规定扩展的属性，而这些属性都通过 RDN 的值表现出来。类和属性都必须在一个模式中进行定义而且定义必须唯一。

模式（schema）是一系列描述可以存储在目录服务器中的数据的定义。模式用来规定对现有类和属性进行定义的语法和匹配规则。如果现有的模式文档不适合描述公司的结构，则可以为该企业创建一个自定义的模式文件。一旦创建了自定义的模式文件，接下来就可以把它包含在 OpenLDAP 的配置文件中了。

由于已有的模式文件不一定适合企业的实际情况，因此通常企业都需要一些特定的属性来描述用户和公司的内部系统结构。在这种情况下，就要在自定义的模式文件中新建符合企业实际的类和相应的属性。当创建自定义模式文件时，一定要注意类和属性的名称必须是唯一的。

为保证类和属性的名称的唯一性，一个很好的策略是给所有新建的类和属性名加上一个前缀。例如，示例企业的一个“`active`”属性的名称可以定义为 `exampleActive`。然后，就可以通过设置 `exampleActive` 的值为 `TRUE` 或 `FALSE` 来启用或禁用这个节点。

此例在 LDAP 中的一个节点如下所示。

```
dn:uid=user1,ou=People,dc=example,dc=com
uid:user1
exampleActive:TRUE
```

一旦这个属性添加到了 LDAP 的一个节点上，就可以在 LDAP 目录树中使用过滤器搜索所有“`exampleActive = TRUE`”的节点，这样就大大提高了对活动用户的搜索速度，当然这只是对于如何使用自定义模式定义的一个示例，要达到以上目的还有很多其他的方法。

■注：在“<http://www.openldap.org/doc/admin24/schema.html>”可以找到 OpenLDAP 管理员手册，里面有关于怎样创建模式文件的内容。

OpenLDAP 默认采用 Berkeley Database 数据库（最初指的是 Berkeley DB，后来又叫 Sleepycat DB，目前由 Oracle 公司拥有，还是原来的名字）。它可以管理包含上百万条记录的数据库，同时访问速度极快且具有良好的可扩展性。Berkeley Database 还针对数据库的读取、搜索以及浏览进行了优化。如果有必要，OpenLDAP 也可以使用其他后端数据库。

■注：可以在以下网址找到更多 Berkeley Database 相关的内容，见“<http://www.oracle.com/technology/documentation/berkeley-db/db/cxx/index.html>”。

如果不满意 OpenLDAP 的表现，还有其他一些不同的目录服务器可以使用。Red Hat 也有自己的目录服务器，在 Fedora 中是免费的，而在 RHEL 中却需要购买使用。微软的活动目录是另外一种 LDAP 服务器，其他还有 Sun 公司的免费目录服务器企业版以及 Novell 公司的 eDirectory。

16.2 总 则

Ubuntu 和 Red Hat 分别提供不同的 OpenLDAP 发行版。Ubuntu 使用最新的 2.4 版，Red Hat 则使用 2.3 版，这两个版本都被认为是稳定和安全的，只是 2.4 版有了一些改进。下面是 OpenLDAP 更新记录列表中给出的各版本之间的不同之处。

OpenLDAP 2.3 版（2005 年 6 月发行）功能增强和可扩展性提高表现在以下方面。

- 扩展了访问控制，包括“出错不提示”等条款。
- 后端配置（cn = cnofig）。
- 密码策略覆盖（Password policy overly）（进行中）。
- 同步提供程序覆盖。
- 支持 Delta-syncrepl。
- LDAP v3 扩展。
 - DIT 管理扩展（进行中）。
 - 组件匹配（实验中）。

OpenLDAP 2.4 版（2007 年 8 月发布）功能增强和可扩展性提高表现在以下方面。

- 升级了 slapd 调度程序。
- 镜像模式和多宿主复制。
- 代理同步复制。
- 扩展的监视功能。
- 多重新覆盖。
- 扩展的记录功能。

- 新的后端套接字（实验中）。
- LDAP v3 扩展。
 - LDAP 链操作支持。
 - LDAP 不使用复制控制支持。
 - LDAP 动态目录服务（RFC 2598）。

在目前阶段读者可能不会全部理解上述这些特性，其实为了让 OpenLDAP 工作起来也没有必要全部理解。不过，考虑到安装 LDAP 的目的，可能会希望选择 Ubuntu 提供的版本而不是 Red Hat 的，因为前者具有比后者更新的特性。

■注：Fedora 中自带有 OpenLDAP 的最新版本。

如果希望使用带多宿主复制功能（也就是具有多个 LDAP 主目录服务器的功能）的版本，那就应该选择 Ubuntu 提供的版本，多宿主功能增强了所安装 LDAP 的冗余度。

还可以应用覆盖特性。覆盖使 OpenLDAP 具有了修整和扩展常规 LDAP 行为的高级功能，如覆盖特性中的 `ppolicy` 覆盖允许使用在 OpenLDAP 原始代码中所没有的密码控制功能。`ppolicy` 覆盖允许进行如密码时限以及最小密码字符长度等设置。

还需要确定支持在企业中使用的哪种类型的认证机制。OpenLDAP 支持两种类型的认证机制：简单和 SASL。简单认证机制有 3 种操作模式。

- 匿名：不提供用户名和密码。
- 无认证：提供用户名，不提供密码。
- 用户名/密码认证：必须提供有效的用户名和密码。

关于 SASL 认证机制，OpenLDAP 管理员指南指出，必须安装一个 Cyrus SASL 并使其运行才能支持 SASL 机制。其实这并不完全正确，是否需要一个正在运行的 Cyrus SASL 取决于想要应用的 SASL 认证机制。在 Cyrus SASL 没有运行时可以很简单地设置 PLAIN/LOGIN 和 DIGESTMD5 机制。不过，安装 Cyrus SASL 还是必需的。SASL 支持如下认证机制。

- PLAIN/LOGIN。
- DIGESTMD5。
- GSSAPI（Kerberos v5）。
- EXTERNAL（X.509 公共/私有密钥认证）。

■注：SASL（PLAIN/LOGIN 和 DIGESTMD5）要求在 `userPasswd` 属性中使用明码密码，关于这样做是否利于安全一直争论不休。其中争论一方这样认为，“一旦进入了数据库，也就可以访问里面的所有密码了。是否加密 `userPasswd` 的值已经没有意义了。”而另一方却认为，“如果进入了数据库就 Game over 了是没错，但至少没有在线传输过明码的密码字符。”

可以到 OpenLDAP 管理员指南的以下页面了解更多不同认证方式的内容。

- <http://www.openldap.org/doc/admin24/security.html#authentication%20Methods>。
- <http://www.openldap.org/doc/admin24/sasl.html>。

16.3 部署

在了解如何将 OpenLDAP 安装到示例系统中之前，先简单了解一下几个部署服务的细节问题。

- 不能将 OpenLDAP 服务安装在已经安装了 Zimbra 邮件服务的主机上，因为 Zimbra 提供了自己的 OpenLDAP 服务，这会与要安装的服务发生冲突。
- 在 DNS 服务器上设置一条 CNAME 记录，使“ldap.example.com”指向“headoffice.example.com”，或者在 DNS 服务器上设置一些 A 记录，指向要安装 LDAP 服务的主机。DNS 设置命令请参考第 9 章。
- 不使用任何目录服务器副本。服务器副本就是在网络中可以存在不止一个 LDAP 服务器共享全部或部分主 LDAP 服务器的数据并响应用户的请求。需要进行附加的设置才可以启用服务器副本。

下面简单了解一下网络情况。假设网络中有一台 Web 服务器，希望设定只有公司中特定组中的用户才有权访问它。通常，需要在网站上增加一个复杂的登录机制，还要有某种形式的用户数据库来保存相关信息，等等。在 Apache 网络服务器上，可以借助 Apache 的 LDAP 模块使 Web 服务器使用 LDAP 服务认证用户的请求，没有通过认证的将不能访问网站。在 OpenLDAP 服务之上还可以配置其他的认证服务。图 16-2 显示了 Web 服务的认证过程。

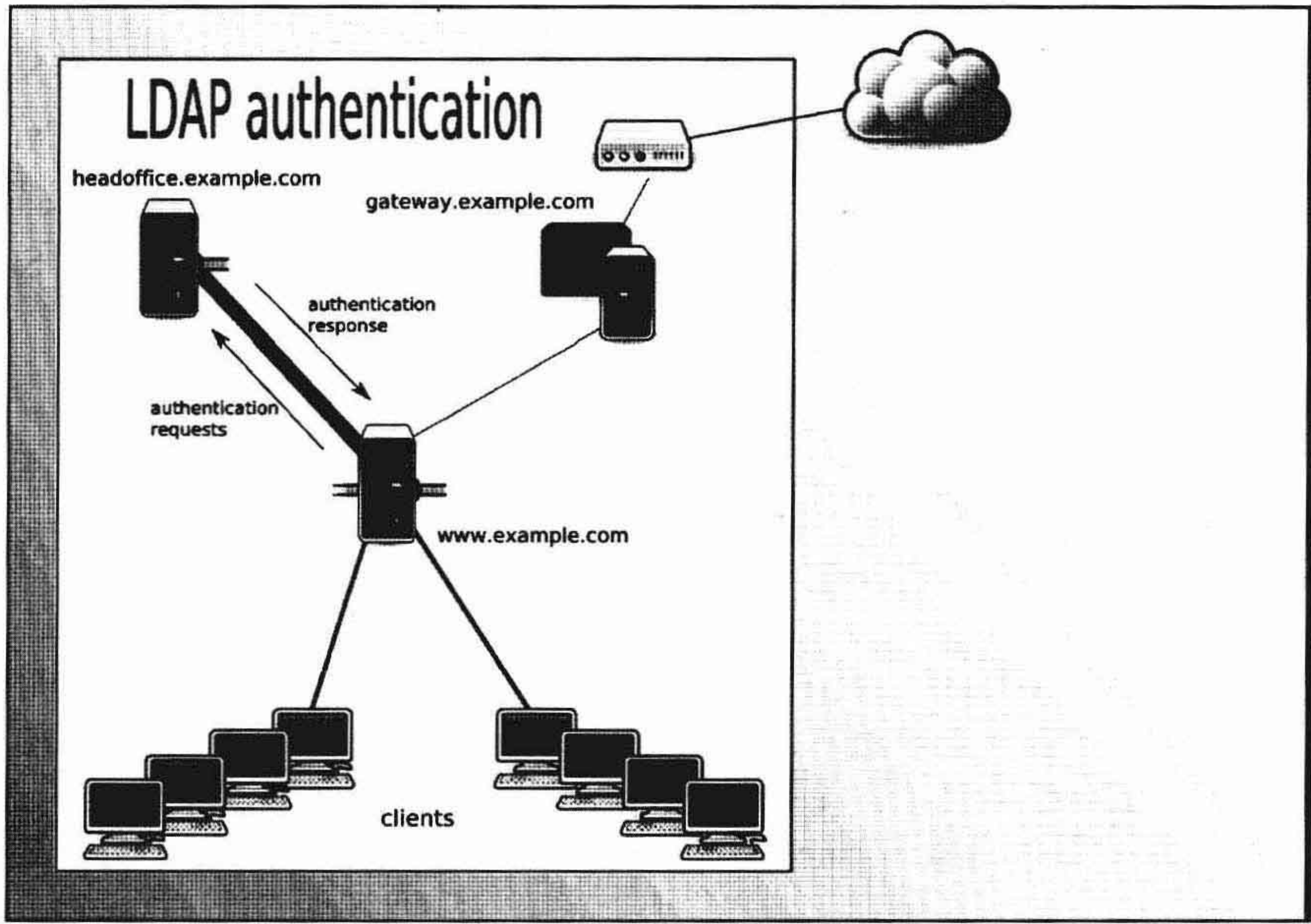


图 16-2 使用 LDAP 进行 Web 服务认证

图 16-2 使用一个简单图标描述了 Web 服务器如何使用 LDAP 来认证桌面用户和网络客户端用户使用 Web 服务（如果硬件资源不充足，LDAP 服务和 Web 服务可以安装在同一台主

机上)。当站点收到访问请求时，提出申请的用户在被允许访问之前要先申请授权。认证申请被传到位于“headoffice.example.com”的LDAP服务上。如果是有效用户，LDAP服务器给Web服务返回一个正确响应，然后用户就可以访问网站了。

本书中介绍的很多服务都可以应用LDAP服务进行认证，这非常有利于把认证服务集中在一个主机上进行管理，减少复杂性，增加认证安全性，而且还可以提供一个保存所有员工详细信息的中心仓库。

下面介绍如何配置LDAP服务以及如何为Apache Web服务配置认证服务。

16.4 安 装

在Red Hat、Ubuntu以及它们的在线工具库中都可以找到OpenLDAP。再次声明，OpenLDAP的两个发行版中存在些微的不同，下面将详细解释这些不同之处。

16.4.1 Red Hat 安装指导

下面介绍在Red Hat主机上安装OpenLDAP服务器的过程。Red Hat网站上提供OpenLDAP的二进制安装包，可以通过使用yum命令或软件包管理界面工具两种方式进行安装。下面是使用yum命令安装的过程。

```
$ sudo yum install openldap openldap-clients openldap-servers
openldap-servers-overlays
```

这样就安装了配置、运行、管理LDAP服务器所必须的文件。Openldap软件包中包含允许主机整合OpenLDAP服务器所需的基础代码。openldap-clients软件包中包含管理和查询LDAP服务器的工具代码。openldap-servers软件包中包含运行OpenLDAP服务器所必需的文件代码。openldap-servers-overlays提供前面所提到的扩展功能。

OpenLDAP的配置文件保存在/etc/openldap/中。

```
$ sudo ls -l /etc/openldap/
total 40
drwxr-xr-x 2 root root 4096 Nov 6 18:57 cacerts
-rw-r----- 1 root ldap 921 Nov 6 18:56 DB_CONFIG.example
-rw-r--r-- 1 root root 327 Sep 14 09:10 ldap.conf
drwxr-xr-x 3 root root 4096 Mar 11 06:02 schema
-rw-r----- 1 root ldap 3729 Nov 6 18:57 slapd.conf
```

专为配置LDAP客户端的文件在/etc/openldap/ldap.conf下。在Red Hat中通常使用system-auth-config工具进行配置，配置时要填写所要求的所有细节信息。

16.4.2 Ubuntu 安装指导

在Ubuntu主机上安装OpenLDAP时，需要安装ldap-utils软件包和slapd软件包，如下命令所示。


```
$ sudo aptitude install ldap-utils slapd
```

执行这个命令的过程中，`slapd` 软件包会要求提供 LDAP 的 `root` 用户的密码。输入一个密码继续安装就可以了。如果不想提供密码，只需按两次 `Enter` 键就可以了（下面的章节中会介绍如何创建密码）。安装完成后，`ldap-utils` 软件包会把管理和查找 LDAP 目录的文件安装到主机中，而 `slapd` 软件包则安装运行和配置 LDAP 目录所必须的文件。客户端配置软件包是 `auth-client-config`（PAM 和 NSS 模式切换）和 `ldap-auth-client`（包含 `ldap-auth-config` 在内的 LDAP 认证套件）。

16.5 配 置

首先来看一看 OpenLDAP 主配置文件，即 `slapd.conf`。接下来介绍如何安全地配置一个 LDAP 目录服务。在示例中，配置的是 Ubuntu 主机中的目录服务，有些目录路径和 Red Hat 中的不尽相同。在 Red Hat 主机中，配置目录在 `/etc/openldap`，不同于 Ubuntu 主机中的 `/etc/ldap`。两种发行版中数据库都保存在 `/var/lib/ldap` 中。

```
#Global Directives:
```

```
#Schema and objectClass definations
include      /etc/openldap/schema/core.schema
include      /etc/openldap/schema/cosine.schema
include      /etc/openldap/schema/nis.schema
include      /etc/openldap/schema/inetorgperson.schema
include      /etc/ldap/schema/ppolicy.schema
include      /etc/ldap/schema/example.com.schema
```

先来看第一个 `include` 语句。OpenLDAP 使用 `include` 后跟文件名的方式把文件内容引入配置文件中。这里引入了 OpenLDAP 提供的通用模式文件和 `example.com.schema` 文件，后者用来引入为特定需要而自定义的属性。在“创建模式”部分会有更多关于模式的介绍。

```
pidfile      /var/run/slapd/slapd.pid
argsfile      /var/run/slapd/slapd.args
```

接下来是 `pidfile` 的位置以及包含 `slapd` 服务器启动文件的位置。例如，`argsfile` 可能包含如下信息。

```
jsmith@ldap:/etc/ldap$ cat /var/run/slapd/slapd.args
/usr/sbin/slapd -h ldap://ldap.example.com ldaps://ldap.example.com ➡
-g openldap -u openldap -f /etc/ldap/slapd.conf
```

日志等级对于安装调试有很重要的作用。实际上，新用户对于日志中都记录了些什么内容可能是比较糊涂的。然而，日志等级是相加的结果，可以在日志中找到非常详尽的细节信息。在制作环境中，建议将此值设为 0，如果愿意，可以使用叠加审核来监控安装过程中所发生的一切。

```
LogLevel      480
```

这里把日志等级设为 480，这时在日志中将会记录搜索筛选器、配置文件处理、访问控

制以及连接信息等内容。表 16-1 中列出了所有可用的日志等级。

表 16-1 附加的日志等级

等 级	描 述
-1	开启所有调试信息。在调高日志等级之前有助于发现 LDAP 服务器的错误
0	关闭所有调试信息。建议实际应用时使用
1	(0x1 跟踪) 跟踪功能调用日志
2	(0x2 数据包) 调试数据包处理日志
4	(0x4 参数) 提供深度追踪调试 (功能参数) 日志
8	(0x8 连接) 提供连接管理日志
16	(0x10 BER) 打印发送或收到的数据
32	(0x20 过滤器) 提供搜索过滤处理日志
64	(0x40 配置) 提供配置文件管理日志
128	(0x80 ACL) 提供访问控制列表处理日志
256	(0x100 状态) 提供连接、LDAP 操作以及处理结果日志 (推荐使用)
512	(0x200 状态 2) 标记要发送的日志条目状态
1024	(0x400 shell) 打印与 shell 后端通信日志
2048	(0x800 parse) parse 条目
16384	(0x4000 sync) 提供 LDAP 同步复制日志
32768	(0x8000 无) 根据设置的日志等级记录信息

如前所述，loglevel 的设置是相加的结果，也就是说可以通过累加想要记入日志项的 level 值来获得更多的日志内容。读者可能已经看出，以上设置为 480 的 loglevel 就包含了 level. 32 (搜索过滤器)、64 (配置处理)、128 (访问控制列表控制) 以及 256 (链接与 LDAP 操作结果)，这样的设置就能够提供细致的信息，对 LDAP 服务来讲已经足够了。如果运行中出现 问题，可以把 loglevel 换到-1 来开启调试模式，这时将会把所有的信息记入日志。还有一点 需要注意，在 production 模式下一般希望把日志等级设为 0。此外还可以把日志信息等级的 十六进制码放在一行中实现日志等级设置的目的，如上例中，还可以把日志等级设置为 0x20 0x40 0x80 0x100。

接下来是模块部分。在配置中添加各种模块来允许对特定功能的访问。

```
# Where the dynamically loaded modules are stored
modulepath      /usr/lib/ldap
moduleload      back_hdb
moduleload      ppolicy.la
```

“modulepath /usr/lib/ldap” 定义了可以找到的模块的路径。接下来定义了要调用的模块的 名称 “back_hdb”，这就是将要用到的整个 Berkeley Database 和 ppolicy。ppolicy 模块允许使 用密码过期以及其他密码控制特性对数据库中的密码进行更好的管理。

```
# The maximum number of entries that is returned for a search operation
sizelimit 500
tool-threads 1
```


可以设定每次查询所返回结果的最大数目，这里设置的是 500（系统默认）。如果不想进行限制，可以设为 `unlimit`。Tool-threads 指令告诉 slap 守护进程只在一个 CPU 上运行索引程序。如果有多个 CPU，也可以设成更大的数字，但是不要大于 CPU 的总数。还可以开启诸如线程、时间限制以及套接字缓存等高级设置。

接下来，就可以配置 TLS 设置了。TLS，或称为传输层安全（Transport Layer Security），用来对服务器和客户端之间的通信进行加密。

```
TLSCACertificateFile /etc/ssl/certs/ca-bundle.crt
TLSCertificateFile /etc/ssl/certs/ldap.pem
TLSCertificateKeyFile /etc/ssl/certs/ldap.key
TLSVerifyClient never
```

上面使用私有密钥创建了一个证书文件并加入了一些细节设置。在 `ca-bundle.crt` 中加入了证书颁发文件。此例中，由于不希望针对某个特定客户端提供的证书文件进行验证，还把 `ILSVerifyClient` 设为 `never`。

下面是安全策略的第二部分，即安全强度因子，简称为 `ssf`。可以为允许访问的链接设置最小的安全强度同时为比较敏感的角色设定较高的通信安全强度。`security` 行描述了特定链接类型的所需安全因子。

```
security ssf=1 update_ssf=112 simple_bind=56
```

“`ssf = 1`”的设置描述了服务器所需的整体安全强度因子。“`Update_ssf = 112`”的设置描述了目录更新所需的整体安全强度因子，“`simple_bind = 56`”是 `simple_bind` 操作所需的安全强度因子。`ssf` 的值如下所示。

- 0 表示没有安全保护。
- 1 表示只有内部保护。
- 56 允许 DES 或其他简单密码。
- 112 允许三重 DES 和其他复杂密码。
- 128 允许 RC4、Blowfish 以及其他现代复杂密码。

系统默认为 0，此设置管理用于连接目录服务的安全策略强度，如 SSL。依据链接的安全强度，可以把安全策略和访问控制列表结合在一起来控制可访问的内容的连接。在本章后面的“访问控制列表”部分有对此更深入的介绍。

接下来要声明保存 DIT 的后端数据库。有几种可选的方式，一般来讲，系统默认为数据库选择 `bdb` 或 `hdb`。还可以从用来代理 LDAP 服务器的类型（`ldap`、`ldif`、`metadirectory`、`perl` 等）中选择一个。

■注：更多有关可选的后端数据库信息，请参考在线文档“<http://www.openldap.org/doc/admin24/backends.html>”。

```
backend      hdb
database     hdb
```

上面的设置选择了 Ubuntu 默认的数据库（`hdb`）作为后端数据库，通常都会选择 `bdb` 或 `hdb`。这两种类型的数据库都是指 Oracle Berkeley DB，也就是前面提到过的最初由 Berkeley

大学（沿用了学校的名称）开发和设计、以前称为 Sleepycat DB 的数据库。它们使用相同的操作命令，唯一不同的是其中一个具有层次布局（hdb），这种层次布局允许子树重命名，其中子树是 DIT 根的分支。

Databae 指令标识着一个新的数据库实例声明的开始。可以声明多个数据库实例。所需定义的下一个细节问题是 DIT 顶端和一个具有完全访问权的用户，比如 root 用户。

```
suffix          "dc=example,dc=com"
rootdn          "cn=admin,dc=example,dc=com"
rootpw          {SSHA}4Tx/MG/x4/mKCf7OheEohw+axXV6lJg+
```

suffix 指令将查询 “dc = example, dc = com” 的结果放入此数据库实例。可以声明多条 **suffix**。**rootdn** 是 root 用户，对数据库有完全访问权，其密码在 **rootpw** 中声明。可以使用如下的 **slappasswd** 命令创建一个密码。

```
$sudo slappasswd
```

输入的密码会被复制并粘贴到 **slapd.conf** 文件中的 **rootpw** 下。下一个指令声明了数据库工作区的路径。

```
directory       "/var/lib/ldap"
```

这个目录对运行 **slapd** 守护进程的用户必须是可写的。在 Ubuntu 中是 **openldap** 用户，在 Red Hat 中是 **ldap** 用户。

下面声明一个覆盖。就像前面提到的一样，覆盖提供 OpenLDAP 服务器中通常不提供的某些附加的功能。此例中，定义一个 **ppolicy** 覆盖。

```
overlay ppolicy
ppolicy_default "cn=default,ou=Policies,dc=example,dc=com"
ppolicy_use_lockout
```

ppolicy 覆盖提供在 LDAP 服务器上更好的控制密码的某些功能。OpenLDAP 自身不具有诸如密码过期和密码历史等密码管理特性，覆盖允许自定义一个策略或者将其他各种策略结合到 DIT 树中的不同部分。下面使用 DN “cn = default,ou = Policies,dc = example,dc = com” 声明了一个默认策略，同时还声明了希望使用 **ppolicy** 中的闭锁（lockout）特性，这个特性允许向被禁止访问的用户返回一条信息。由于这也会向攻击者提供用户名是否存在的信息，读者可能会想关闭这个特性。

在“密码策略覆盖”一章中，把 LDIF 装载到 OpenLDAP 后，会在 **ppolicy** 覆盖中添加更多内容。

■注：LDIF 是声明 LDAP 数据库内容的文件格式。在“LDIF 与添加用户”一章中会有如何通过客户端工具创建和使用 LDIF 文件的介绍。

下面还要声明数据库性能调整方面的条目。如果 **slapd** 启动时没有已存在的配置文件，可以在 **DB_CONFIG** 文件中使用 **dbconfig** 对数据库进行各方面的设置。如果有配置文件，所做的新设置将会被忽略，这时应该直接编辑位于 **/var/lib/ldap** 路径下的 **DB_CONFIG** 文件。

```
dbconfig set_cachesize 0 5242880 0
dbconfig set_lk_max_objects 1500
```



```
dbconfig set_lk_max_locks 1500
dbconfig set_lk_max_lockers 1500
```

以上对数据库的调整参数中使用 `set_cachesize` 项设置了 512MB 的高速缓存。`set_cachesize` 项的语法格式是首先为千兆字节数 (0)，接着是字节数 (5242880)，最后是高速缓存分段数量 (0)。Dbconfig 的其他设置都是标准默认的，至少在 Ubuntu 中是这样的。

注：可以在下述网址找到关于 dbconfig 设置相关的更多内容，见 “<http://www.oracle.com/technology/documentation/berkeley-db/db/ref/toc.html>”。

下面设置索引。索引用来加速数据库搜索。

```
index    objectClass eq
index    cn sub
index    exampleActive pres,eq
index    uid,uidNumber pres,eq
```

通常，应该索引用户经常搜索的内容。为了在 e-mail 用户地址簿中寻找用户姓名时定位在地址簿中的分录位置，通常需要搜索通用名，或者说 `cn`。这种情况下，就需要有一个子字符串或 `sub` 优化对 `cn` 属性的索引。表 16-2 列出了可用的常见索引类型。

表 16-2 常见索引类型

类型	说明
sub	用于优化含有通配符*的字符串搜索，如 <code>cn = Jane*</code>
eq	用于优化字符串的精确搜索，如 <code>sn = Smith</code>
pres	用于优化对象类或属性的搜索，如 <code>objectclass = person</code>
approx	用于优化模糊搜索，如 <code>sn~ = Smi*</code>

还有其他的索引类型，可以通过对 `slapd.conf` 文件使用 `man` 命令查看。通常情况下希望对 `objectclass`、`cn` 以及 `uid` 字段进行索引，因为这些字段是用户尝试认证时经常会搜索的字段。

另外两个与 Berkeley DB 性能相关的设置是 `lastmod` 和 `checkpoint`。

```
lastmod    on
checkpoint 512 30
```

`checkpoint` 项告诉 `slapd` 守护进程将缓存内容写入磁盘，同时在事务日志中写入一个检查点。此项设置显示希望每过 30 分钟或每当事务日志满 512KB 大小时检查点进程运行一次。`lastmod` 项记录 RDN “`cn = lastmod,dc = example,dc = com`” 的最后一次修改。修改项目的 DN、修改类型以及何时修改都会被记录下来。

使用 `include` 项声明在什么位置可以找到访问控制列表。在“访问控制列表”部分将讨论访问控制列表及其使用的内容。

```
include    /etc/ldap/access.example.com
```

下面使用 `database` 指令创建另一个数据库。这个 `config` 类型的数据库用来保存 `slapd` 的配置。可以使用它更改 LDAP 数据库而不用重启服务器。


```
#####
database config
rootdn cn=admin,cn=config
rootpw {SSHA}4Tx/MG/x4/mKCf7OheEohw+axXV6lJg+
```

这个数据库需要一个根 DN 和一个密码。当 slapd 守护进程从 slapd.conf 文件和其他包含文件中启动时这个配置数据库就创建了。slapd.conf 文件中的指令在 slapd.d 目录下被转换为 LDIF 格式的文件，依据所使用发行版的不同，slapd.d 目录一般在路径/etc/ldap 或/etc/openldap 下。此时就可以使用 LDAP 工具管理数据库了。

■注：以下地址有关于配置引擎数据库的更多内容，见“<http://www.openldap.org/doc/admin24/slapdconf2.html>”。

到现在为止，我们还没有准备好启动 LDAP 服务器。首先要创建前面在 slapd.conf 文件中所提到的附加文件，还要创建模式文件“example.com.schema”和访问控制列表文件“access.example.com”。

16.5.1 创建模式

slapd.conf 文件中声明了一个想要添加的名为 example.com.schema 的模式文件。在这个 schema 文件中，将会引入一个类以及用来判别一个用户账号是否激活的属性。首先看一下如何在模式文件中声明一个类。以下是 schema 文件夹中 core.schema 文件的内容。

```
objectclass ( 1.3.6.1.4.1.1466.344 NAME 'dcObject'
  DESC 'RFC2247: domain component object'
  SUP top AUXILIARY MUST dc )
```

这个对象类是将被引入 DIT 中的主要对象类中的一个，因为要声明所用实体的类型，所以从 objectclass（细节）开始。声明对象和属性时要特别注意其中空格的使用。声明中的数字“1.3.6.2.4.1.1466.344”就是个人企业序号（PEN），或者叫对象标识符（OID），这组唯一的序号用来识别不同的对象，接触过 SNMP 的读者应该对此不陌生。

■注：可以到互联网编号分配管理机构（IANA）注册一个自己的 OID 或 PEN，机构网址是“<http://pen.iana.org/pen/penapplication.page>”。

然后给对象类一个名称“dcobject”以及相关描述。下面一行显示它继承自对象类 SUP top。关键字 AUXILIARY 表示对象类的类型。MUST dc 表示如果此对象是在目录服务器中声明的，那就必须给它定义一个 dc 属性项。

■注：在“<http://www.rfc-editor.org/rfc/rfc4512.txt>”中有关于声明对象类所有细节描述的 RFC。在“<http://www.openldap.org/doc/admin24/schema.html>”可以快速浏览到模式扩展的有关内容。

下面来整体看一下模式文件“/etc/ldap/schema/example.com.schema”。

```
# $Id$
attributetype ( 1.1.3.10 NAME 'exampleActive'
```



```
DESC 'Example User Active'  
EQUALITY booleanMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 )
```

```
objectclass ( 1.1.1.2 NAME 'exampleClient'  
  SUP top AUXILIARY  
  DESC 'Example.com User objectclass'  
  MAY ( exampleActive ))
```

上述两个模式类定义了两个 OID，都是随意编的。它们可能和已有的模式文件发生冲突。要想避免这种情况的发生，就必须按部就班地申请自己地 PEN。假设申请到了一个编号为“1.3.6.1.4.1.11111”的 OID，其中“1.3.6.1.4.1”是一个 IANA 的 arc，或叫节点，“11111”是区别于其他企业的唯一的编号，这样就可以在上述的模式文件中的相应位置使用自己的 OID 了。

■ **特别注意：**如前所述，OID 编号为“1.3.6.1.4.1.11111”是为了演示而随意编写的。请不要在实际应用的环境中随意编写或应用 OID。应该实际申请一个 PEN，否则会有冲突或数据破坏的危险。关于 OID 和 LDAP 更多的信息，请访问如下网址“<http://www.zytrax.com/books/ldap/apa/oid.html>”。

```
attributetype ( 1.3.6.1.4.1.11111.3.1.1 NAME 'exampleActive'  
  DESC 'Example User Active'  
  SINGLE-VALUE  
  EQUALITY booleanMatch  
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 )  
  
objectclass ( 1.3.6.1.4.1.11111.3.2.1 NAME 'exampleClient' SUP top AUXILIARY  
  DESC 'Example.com User objectclass'  
  MAY ( exampleActive ))
```

拥有了一个 PEN 或 OID 后，可以把它分成几个有用的段（也称为节点或 arc）。通常，一个 OID 不仅可以用于 LDAP 的模式对象，还可以用于 SNMP 中的 MIB。如前所述，序号“1.3.6.1.4.1.11111.3”已经被分出来定义 LDAP 模式。下面，就可以把所有的对象类定义在“1.3.6.1.4.1.11111.3.2”之下，同时在“1.3.6.1.4.1.11111.3.1”下定义所有的属性。

■ **注：**把“1.3.6.1.4.1.11111.3.1”和“1.3.6.1.4.1.11111.3.2”分配给 LDAP 的类和属性定义是完全随机的。可以给模式定义选择任何一个想用的数字。

将 exampleActive 属性定义为布尔量匹配，意味着它可以有真、假两个值。值为真时说明账号是激活的，值为假时说明账号没有激活。可以对此属性进行索引，以提高搜索的速度。这就是为什么要在前面的 slapd.conf 文件中作如下设置。

```
index      exampleActive pres,eq
```

exampleClient 对象类定义显示在把上面的对象类引入 DN 节点时可能已经包含了一个 exampleActive 属性（由 MAY 所标识的）。如果强制其存在，可以使用 MUST，这是一个 AUXILIARY 类型的类并具有由 SUP top 定义的高级类，默认类的类型为 STRUCTURAL。在节点中必须含有一个 STRUCTURAL 类型的类，但是不能将两个 STRUCTURAL 类型的类指向同一个父类或高级类。

■注：对于上述问题的解释，请参考网址：“<http://www.linuxlaboratory.org/?q=node/44>”。介绍 LDAP 模式文件的 RFC 可以在以下网址找到“<http://www.rfc-editor.org/rfc/rfc4512.txt>”。

在本章的“LDIF 与添加用户”部分声明用户时将会用到 `example.com.schema` 文件。

16.5.2 访问控制列表

如果想让 LDAP 服务器安全，必须针对目录树的各个部分为每一个服务器访问连接分配一个指定的入口。可以从以下几方面进行划分：允许何处的连接，安全等级，或对需要获得授权才能访问的连接进行加密直到允许访问的分支或属性等。还可以通过以下几个访问等级对连接请求进行授权：管理，写，读，搜索，认证。

已经使用如下指令把配置信息引入到了 `slapd.conf` 文件中。

```
include /etc/ldap/access.example.com
```

可以通过以下最基本方式的语法对访问进行授权。

```
access to what [ by who [ access ] [ control ] ]+
```

what 是指在 LDAP 数据库中的一个实体，**who** 是指请求信息的客户端，还有 **access** 是指给予客户端此次访问的等级。**control** 项规定进入后如何处理列表，此项为可选项。

下面的简单示例中，DIT 中的所有内容都被定义为只读访问。

```
access to *
        by * read stop
```

使用通配符*来表示一般无限制的访问，上述访问控制规定任何用户对 DIT 中的任何内容都具有只读访问权限。其后的控制语句告诉 `slapd` 停止其他任何指令的运行。访问控制列表中语句的优先级顺序很重要，优先级高的指令先处理，优先级低的指令后处理。当给定一个权限或访问等级时，意味着比它等级低的权限也是允许的。比如，给定了 `read` 权限也就自动允许了它前面的 `disclose`、`auth`、`compare`、`serch` 等访问等级，包括 `read` 访问权限。表 16-3 列出了可以给实体访问请求授权的访问等级。

表 16-3 访问权限

访问等级	权限
none	禁止访问
disclose	禁止访问并返回出错信息
auth	允许绑定访问（需认证）
compare	允许进行实体对比
search	允许对 DIT 某个部分进行搜索
read	只读
write	可写
manage	所有权限，可以删除实体

如果选择 **none**，则禁止对实体的任何访问，同时不让访问请求者返回任何出错信息，这样能防止泄露诸如 DIT 上有什么和没有什么的信息。而 **disclose** 访问等级，不像 **none**，会给请求客户端返回一个出错信息。

想要进一步了解对实体的访问申请，就需要知道申请访问者是谁。可能声明了很多个 **who**，每个都使用一个特定的关键字，这些关键字可能带有类似 **regex** 或 **exact** 的类型（**style**）限定符。其中 **Regex** 类型指的是用来匹配 DN 不同部分的正则表达式。当然，进行这样的处理，就要付出更高的代价来运行访问控制列表。

■注：关于正则表达式的提示信息可以参考 OpenLDAP 管理员指南，见 “<http://www.openldap.org/doc/admin24/access-control.html#Tips%20for%20using%20regular%20expressions%20in%20Access%20Control>”。

通常在一个项目中，对可以访问的内容及谁可以访问描述得越精确，其运行代价越小。比如：

```
access to dn.subtree=ou=People,dc=example,dc=com
    by dn.exact="cn=admin,ou=meta,dc=example,dc=com" read
```

这里再次将组织中 **People** 节点下的所有内容置为只读访问，而且还明确界定了此访问权限只赋予节点 DN “**cn = admin,ou = meta,dc = example,dc = com**”。

定义授权访问的内容可能会很棘手。有几种进行授权访问的通用方法，如下所示。

```
dn.base
dn.one
dn.subtree
db.children
```

为了解释这些是如何涉及正在使用的对象的，这里从 OpenLDAP 管理员指南中借鉴一个实例。假设有如下所示的一个列表。

```
0: dc=example,dc=com
1: cn=Manager,dc=example,dc=com
2: ou=people,dc=example,dc=com
3: uid=jsmith,ou=people,dc=example,dc=com
4: cn=addresses,uid=jsmith,ou=people,dc=example,dc=com
5: uid=ataylor,ou=people,dc=example,dc=com
```

当需要在 DIT 下的某些部分进行工作时，可以声明访问等级模式匹配的范围。

```
dn.base="ou=people,dc=example,dc=com" match 2;
dn.one="ou=people,dc=example,dc=com" match 3, and 5;
dn.subtree="ou=people,dc=example,dc=com" match 2, 3, 4, and 5; and
dn.children="ou=people,dc=example,dc=com" match 3, 4, and 5.
```

声明的范围将会对应 DIT 树中的相应部分。正如所看到的，**dn.base** 所指定的范围将会只参考声明树中 “**ou = people, dc = example, dc = com**” 部分的访问等级。**dn.one** 所指定的范围将会在紧接着 “**ou = people, dc = example, dc = com**” 的等级下运行。**dn.subtree** 所指定的范围将会在包括 “**ou = people, dc = example, dc = com**” 在内的其下所有等级下运行，而 **dn.children** 将会在 “**ou = people, dc = example, dc = com**” 之下的所有等级运行。

■注：关于将权限赋予谁的问题可以参考“<http://www.openldap.org/doc/admin24/access-control.html#Who%20to%20grant%20access%20to>”。

在 LDAP 中可以使用过滤器，这是剔除不良数据、保留所需精确结果的一种手段。在访问控制列表中，也可以使用过滤器使授权访问的内容更加具体。

```
access to dn.subtree="ou=people,dc=example,dc=com" attrs="userPassword"
    by dn.exact="cn=admin,ou=meta,dc=example,dc=com" write
    by * none
```

在这个例子中，声明将此条规则应用于包括节点“ou = people, dc = example, dc = com”在内的以下所有节点以及在这些节点中可以找到的所有名称为 userPassword 的属性。此例中，属性 userPassword 就是过滤器。管理员用户被赋予对 userPassword 属性的可写权限，其他任何操作都将被默认拒绝。

man 页面是参考更多访问控制列表相关信息的很好的资源，还有 OpenLDAP 管理员指南也很不错，见“<http://www.openldap.org/doc/admin24/access-control.html>”。

下面简单梳理一遍将要在名为 example.com 的 LDAP DIT 中使用的访问控制列表。首先定义了对密码信息的访问权限。就像前面所提过的，访问控制列表的读取和执行都是从上到下的，因此把比较敏感的访问控制信息放在顶部非常重要，这样它们就不会被更高级别的条目所覆盖。

```
access to attrs=userPassword,shadowLastChange,entry
    by ssf=128 dn.exact="cn=Webadmin,ou=meta,dc=example,dc=com" auth
    by ssf=128 anonymous auth
    by ssf=128 self write
```

这一部分中，对用户密码信息的访问进行了限制，这些信息通过 DIT 保存在 userPassword 和 shadowLastChange 属性中，通常在“ou = people, dc = example, dc = com”分支下。这里将只给予管理员特殊的访问权限。webadmin 用户将被用来将 LDAP 服务器绑定到 Web 服务器，以便于 Web 用户的授权认证，而且只允许安全强度系数大于或等于 128 的链接访问这些属性。

■注：前面提到过安全强度系数，也可以使用其他选项来限制对属性的访问，比如提供一个 peername 或配置从何处接受访问等。对此以及访问控制列表的更多通用信息，请参考“<http://www.openldap.org/doc/admin24/access-control.html>”。

授权 anonymous auth 的访问权限，也就是，客户端无需绑定到 LDAP 服务器来进行认证。这对于单点登录服务非常重要，本章后面的“单点登录”部分还会对此进行详细介绍。还可以通过允许 self 的写权限使用户可以修改自己的密码。

如前所述，顺序很重要。当一个访问请求来到 LDAP 主机时，访问控制列表就要进行检索，如果找到匹配的内容，访问要么被允许，要么被禁止。可以通过把访问控制列表按从最多访问请求到最少访问请求的顺序进行排列来加快访问速度。通常希望最常见的访问请求靠近访问控制列表的顶部，而最不常见的靠近底部。假设实例中的一些 meta users 将会对目录服务器中的很多不同部分进行访问，那么这些用户就会产生最常见的访问请求。这就是为什

么要把控制列表中对 **meta users** 组的处理放在列表的顶部且紧靠着用户密码的条目。

“**ou = meta**”分支中包含了对目录服务器进行访问控制的用户。通常不要求用户绑定到目录服务器，但有时也希望他们绑定，比如运行 Web 验证的时候。前面把对用户密码条目的访问权限赋予了 **Webadmin** 用户。下面就来定义这些 DN 对自己信息的权限。

```
access to dn.subtree="ou=meta,dc=example,dc=com"
    by group.exact="cn=admins,ou=Groups,dc=example,dc=com" write
    by self read
```

给“**cn = admins**”组赋予了对企业级单元的可写访问权限，同时要把系统管理员用户放在这个组中，并赋予元 **meta users** 对自己信息的只读访问权限。这可以避免用户更改定义在“**ou = meta**”企业级单元下的任何条目，同时给予了这些用户更高的安全性。

下面，对“**ou = People**”分支下所有内容的访问进行授权，切记在访问控制列表的前面部分已经定义了对用户密码进行访问的权限，早期的定义将覆盖在这里对前期已经定义过的属性的任何访问权限。管理员账户至少需要只读访问权限，这里给的是可写的权限，因为希望 **admins** 用户组也能够时时对一些细节进行修改。**webadmin** 用户只需要只读权限。另外还需用 **self** 关键字赋予条目对自己信息的只读访问权限。

```
access to dn.subtree="ou=People,dc=example,dc=com"
    by dn.exact="cn=Webadmin,ou=meta,dc=example,dc=com" read
    by group.exact="cn=admins,ou=Groups,dc=example,dc=com" write
    by self read
```

在特定的网络环境中可能对此有不同的要求，而且 **self** 的访问权限也通常是可写而不是只读。上面的设置只给予了用户对定义个人信息的个人属性细节进行修改的权限，尽管只读的访问权限并不允许这样做。

下面的代码中，对包含组用户信息的“**ou = Groups**”分支进行访问授权。

```
access to dn.subtree="ou=Groups,dc=example,dc=com"
    by dn.exact="cn=Webadmin,ou=meta,dc=example,dc=com" read
    by group.exact="cn=admins,ou=Groups,dc=example,dc=com" write
    by users read
```

正像代码中显示的一样，此设置与“**ou=People**”分支很相似，都有访问权限相同的管理员账户。然而，通过专门定义 **users read** 而允许了认证用户的读权限。

接下来是“**ou = Hosts**”企业级单元，有人也称之为 **machines**，称呼是什么都行。这个单元包含了所有的主机信息、IP 地址信息、位置信息等。前面已经用过了子树的范围，并且授予“**cn = admins**”以外所有内容的最小化写权限。

```
access to dn.subtree="ou=Hosts,dc=example.com"
    by group.exact="cn=admins,ou=Groups,dc=example,dc=com" write
    by anonymous read
```

这里，赋予“**cn = admins**”组可写访问权限，给没有建立绑定链接的 **anonymous** 客户端（已经获得了用户名和密码）只读权限。包括 **Samba** 在内的很多应用程序都要用到“**ou = Hosts**”企业级节点。

最后一条规则是一条完全否定规则，它会强制拒绝其他形式的访问。这基本上是多余的，

因为任何没有授权的特定访问都会被拒绝，然而，它是访问控制列表结束的标志，同时它也防止了任何在它后面添加的访问控制规则被错误地读入。

```
access to * by * none stop
```

通配符*匹配任何内容，表示任何形式的访问都会被拒绝，同时所有进一步的访问都会被控制域的 stop 选项所停止。其他两个可用的处理控制符是 break 和 continue。

找到匹配项时，break 控制符将停止当前访问控制组中的下一步操作并跳转到下一个控制组。匹配完成后，continue 控制符将继续当前访问控制组的下一步操作，并允许更高级别的访问授权。stop 选项只是立即停止进一步的操作，此项为系统默认。

完整的访问控制列表如列表 16-1 所示。

列表 16-1 完整的访问控制列表

```
access to attrs=userPassword,shadowLastChange,entry
  by ssf=128 dn.exact="cn=Webadmin,ou=meta,dc=example,dc=com" auth
  by ssf=128 anonymous auth
  by ssf=128 self write

access to dn.subtree="ou=meta,dc=example,dc=com"
  by dn.exact="cn=Webadmin,ou=meta,dc=example,dc=com" read
  by group.exact="cn=admin,ou=Groups,dc=example,dc=com" write
  by self read

access to dn.subtree="ou=People,dc=example,dc=com"
  by dn.exact="cn=Webadmin,ou=meta,dc=example,dc=com" read
  by group.exact="cn=admin,ou=Groups,dc=example,dc=com" write
  by self write
  by users read

access to dn.subtree="ou=Groups,dc=example,dc=com"
  by dn.exact="cn=Webadmin,ou=meta,dc=example,dc=com" read
  by group.exact="cn=admin,ou=Groups,dc=example,dc=com" write
  by anonymous read

access to dn.subtree="ou=Hosts,dc=example.com"
  by group.exact="cn=admin,ou=Groups,dc=example,dc=com" write
  by dn.exact="cn=Webadmin,ou=meta,dc=example,dc=com" search

access to *
  by * none
```

有了这些文件，就可以试着启动 LDAP 服务器了。

16.5.3 启动 slapd 守护进程

可以通过两种方式运行 slapd 守护进程：使用 slapd.d 配置引擎或不使用它。正如前面所提到的，配置引擎允许在服务运行时使用标准的 LDAP 正则表达式和命令来更改 LDAP 配置。两种方式都支持，只不过不使用配置引擎的运行方式最终将被放弃，因此还是建议用户使用配置引擎来启动 slapd。不过需要手动启动服务来创建 slapd.d 配置引擎。要启动和创建

slapd.d, 运行以下命令。

```
$ sudo mkdir /etc/ldap/slapd.d
```

在 Red Hat 中, 需要手动创建/etc/openldap 目录。然后, 进入配置目录(/etc/openldap 或/etc/ldap), 执行以下命令。

```
$ sudo slapd -f slapd.conf -F slapd.d -u openldap -g openldap -d -1
```

注意, 此命令在前台运行, 这样就可以看到它尝试开始运行时是否有问题出现。使用“-d -1”选项使其前台运行, “-1”表示完整调试模式的日志等级。在 Red Hat 主机上, 用户使用“-u ldap -g ldap”运行 OpenLDAP, 而在 Ubuntu 主机上使用“-u openldap”。“-f slapd.conf”指定配置文件, -F 指定包含 LDIF 配置引擎文件的 slapd.d 目录。

slapd 实例运行后, 自动在 slapd.d 目录下创建几个文件和目录。这些文件包括在 slapd.conf 以及其他用 LDIF 文件格式引用的文件中指定的 LDAP 设置。

■注: 在 RFC2849 中有对 LDIF 文件格式的描述, 见 “<http://www.ietf.org/rfc/rfc2849.txt>”。

LDIF 由如下所示的基本形式的一个条目组成。

```
# comment
dn: <distinguished name>
<attrdesc>: <attrvalue>
<attrdesc>: <attrvalue>
<blank line>
```

下面来看一下在 slapd.d 目录下创建的众多 LDIF 文件中的一个。

```
cat /etc/ldap/slapd.d/cn\=config.ldif
dn: cn=config
objectClass: olcGlobal
cn: config
olcConfigFile: /etc/ldap/slapd.conf
olcConfigDir: slapd.d
olcArgsFile: /var/run/slapd/slapd.args
olcAttributeOptions: lang-
olcAuthzPolicy:none
olcConcurrency: 0
olcConnMaxPending: 100
olcConnMaxPendingAuth: 1000
olcGentleHUP: FALSE
```

可以看到, 它由一个唯一的名称“dn = config”以及几个属性的声明和值组成。有了这个文件, 就可以使用类似 ldapmodify 一样的 LDAP 工具来改变 OpenLDAP 的配置而不用重启服务器。

■注: 在以下网址可以找到关于管理 OpenLDAP 服务器配置的更多信息, 见 “<https://help.ubuntu.com/8.10/serverguide/c/openldap-server.html>”。

和在主机上运行的其他服务一样, 服务的状态(停止或运行)都受/etc/init.d 文件中的脚本所控制。可以在 Red Hat 中使用以下命令手动启动或停止一个 LDAP 服务器。


```
$ sudo /sbin/service ldap start|stop
```

在 Ubuntu 中则可以使用 `invoke-rc.d` 命令。

```
$ sudo /usr/sbin/invoke-rc.d slapd start|stop
```

服务启动后，就可以使用 `tail` 命令查看日志，以确认服务启动成功。此外，还可以使用日志监控以及解决访问请求过程中出现的问题。

16.5.4 设置 LDAP 客户端

Ubuntu 和 Red Hat 都使用 `ldap.conf` 文件来默认配置整个系统的 LDAP 客户端。使用 OpenLDAP 库的应用程序用这些配置来获取 LDAP 的细节信息。可以在 `/etc/ldap` 下找到 Ubuntu 的配置文件，在 `/etc/openldap` 下找到 Red Hat 的配置文件。

注：重要的是，不要把此处的配置文件和一个名字也是 `ldap.conf` 的 `nss_ldap` 文件混淆起来。两种发行版中，都可以在 `/etc/ldap.conf` 路径下找到此文件。

需要编辑 `ldap.conf` 文件并加入下面几行文本。在实例中，要做点小小的欺骗，但是不需要担心为 LDAP 客户端设置 SSL 认证。如果一个主机要复制 LDAP 服务器，需要绝对保证服务器和客户端都启动了 SSL 认证机制。可以使用 `man` 命令查看 `ldap.conf` 的详细信息。

```
URI ldap://ldap.example.com/
BASE dc=example,dc=com
TLS_CACERT /etc/ssl/certs/ca-bundle.crt
TLS_REQCERT allow
```

URL 指向 LDAP 服务器。BASE 后面是 LDAP 操作的默认基础 DN。TLS_CACERT 指向 CA 认证文件，此文件中包含了 `example.com` 的 CA 认证信息。在 TLS_REQCERT 部分指定的 `allow` 意味着认证信息将会试着被验证，但若验证没有通过，还是会进行连接。其他可用选项还有 `demand`，作用是若没有提供认证信息或认证信息无效对话立即结束（此为系统默认值）；`try`，作用是若没有提供认证信息则连接继续进行，但是若提供了一个无效的认证信息，连接立即结束；`never`，作用是建立连接之前主机不要求或验证服务器认证信息。

如果使用的是 Red Hat 主机，应该会在 `/etc/tls/certs` 目录下找到 SSL CA 认证的相关信息。

16.6 LDAP 管理及其工具

那么如何使用 LDAP 管理条目呢？有很多工具可以完成这项工作。可以使用命令行工具从文本文件中添加条目、搜索已有的条目以及删除条目。所用的文本文件必须遵循 LDIF 文件格式。LDIF 文件的格式如下所示。

```
dn: <dn entry>
objectclass: <objectclass to be included>
attribute: <attribute value described in an objectclass>
```


为所处理的不同部分创建单独的 LDIF 文件通常是个不错的主意。比如，对 “ou = People, dc = example, dc = com” 下所有内容的处理可以放在 people.ldif 文件中，对 “ou = Groups, dc = example, dc = com” 下所有内容的处理可以放在 groups.ldif 文件中。然而，对于新建的 LDAP 服务器，也可以把所有的项目放在一个文件中，但是若 LDAP 服务器已经存在很多条目时，就需要谨慎了，因为如果在其中添加一个已有的条目时将会出错。这种情况下，可以在条目每一行的开始加上 “#” 将其注释掉。可以使用 LDAP 工具通过 “-f filename” 选项引用 LDIF 文件，后续部分会有详细介绍。

另一种管理条目的方式是使用一个可用的 GUI 工具。后面将会在 “LDAP 账户管理：基于网络的 GUI” 部分详述如何安装和配置一个基于网络的 GUI 工具。

16.6.1 LDIF 文件与添加用户

在每一个 DIT 的顶部都是根 DN，DIT 都开始于对 dcObject 类的声明。下面是将用来组成 LDAP 服务器的 LDIF 文本文件的片段。

```
dn: dc=example,dc=com
objectclass: dcObject
dc: example
```

这里声明将创建根 DN “dc = example, dc = com”。依据先前在 slapd.conf 中声明的 core.schema 中的对象类 dcObject，应该引入 dc 属性。再来看一下在 core.schema 文件中的对象类声明。

```
objectclass ( 1.3.6.1.4.1.1466.344 NAME 'dcObject'
    DESC 'RFC2247: domain component object'
    SUP top AUXILIARY MUST dc )
```

可以看到这里是如何使用前面在 “dn:dc = example,dc = com” 中声明的对象类的。因为要使用在对象定义时使用了 MUST 修饰符的 dc 属性，所以必须要在下一部分添加用户时添加这个条目，这应该是添加到 LDAP 服务器中的第一个条目。

接下来，需要配置公司中的用户了，因此要先声明企业级节点 People。如果需要，可以把这一部分作为人员条目单独放在一个新的文件中。

```
dn: ou=People,dc=example,dc=com
objectclass: organizationalUnit
ou: People
```

可以看到，LDIF 格式要求声明 DN，以及要用到的对象类以及属性。每一个声明用空行隔开。声明的顺序也很重要，在企业级节点没有创建之前，不能在 “ou = People,dc = hitwise, dc = com” 下创建用户。对象类 organizationalUnit 要求像这里做的一样声明 ou 属性 “ou:People”。

现在添加一个用户 jsmith。

```
dn: uid=jsmith,ou=People,dc=example,dc=com
objectclass: top
objectclass: person
objectclass: posixAccount
```



```

objectclass: exampleClient
cn: Jane Smith
sn: Smith
uid: jsmith
uidNumber: 1000
gidNumber: 1000
exampleActive: TRUE
homeDirectory: /home/jsmith
userPassword: {SHA}IOq+XWSw4hZ5boNPUtYf0LcDMvw=

```

首先看一下 DN。可以看到使用 uid 属性声明了 DN，也可能已经在这里使用了几个变量来声明而不是“uid = jsmith: cn = Jane Smith”或“mail = jane.smith@example.com”，最终在系统中使用哪种方式取决于使用者认为哪种方式对服务器来讲是最好的（不要忘了创建索引）。接下来，需要声明对象类 top 和 person，是否包含 posixAccount 和 exampleClient 对象类可选。top 对象类（一个超级类）对于支持其他对象类是必须的。person 对象类提供了 sn（别名）和 cn（通用名）属性。对象类 posixAccount 提供了对 Unix/Linux 主机都非常有用的属性，如 userPassword、uid、uidNumber、gidNumber 以及 homeDirctory。exampleClient 对象类提供了 exampleActive 属性，此属性可用来激活或停止用户。请注意，像 exampleActive 一样的布尔属性的属性值必须大写。

现在将添加 organizationalUnit 类型的 Groups，以便于使用用户组管理对用户信息的访问。可以再为用户组创建新的 LDIF 文本文件。

```

dn: ou=Groups,dc=example,dc=com
objectclass: top
objectclass: organizationalUnit
ou: Groups

```

可以看出为 Groups 创建企业级节点和前面声明 People 企业级节点的方法非常相似。就像模式定义所要求的那样，使用并声明了 ou 属性来命名 DN。接下来，声明用来组织管理员用户的 admins 组。

```

dn: cn=admins,ou=Groups,dc=example,dc=com
objectclass: top
objectclass: groupOfNames
cn: admins
member: uid=ataylor,ou=People,dc=example,dc=com

```

上面使用 groupOfNames 对象类声明了一个用户组列表，这个列表中的组只可以添加成员。可能还使用了 posixGroup 对象类，这使得用户也可以使用 gidNumbers。可以在组中想添加多少成员就添加多少，只要简单地在新的一行中加入“member: DN”就可以了。

现可以看看如何在 LDAP 数据库中添加详细信息。为此，将使用 OpenLDAP 自带的 ldapadd 工具。

16.6.2 使用 LDIF 文件添加用户

所有的 LDAP 工具都使用相同的一组选项，可以使用这些选项连接 LDAP 服务器。OpenLDAP 客户端工具可以用来连接其他软件厂商提供的 LDAP 服务器。表 16-4 列出了大多数

LDAP 工具可以使用的通用选项。

表 16-4 LDAP 工具通用选项

选项	描述
-x	执行简单绑定
-v	显示详细输出信息
-W	提示输入密码
-f	指向一个输入文件，工具不同时其类型也可能会不同
-D	指定要绑定的 DN，此 DN 必须对所影响的条目有相应的读写权限
-Z	尝试使用 TLS 进行 LDAP 连接，-ZZ 表示必须在 TLS 应用无误后才能继续连接
-Y	指定连接 LDAP 服务器时的 SASL 认证机制。请确认在 SASL 配置时启用了此选项
-X	指定 SASL 的 authzid，也就是 SASL 绑定的授权请求 ID
-U	指定 SASL 的 authcid，也就是 SASL 绑定的认证 ID
-b	绑定基础 DN。可以指定一个开始的基础 DN，而不用查询整个树
-s	指定搜索范围。可以是基础 DN、分 DN 或子 DN

表中有的选项并不是对所有 LDAP 工具都有效。LDAP 工具的命令语法通常如下所示。

```
ldaptool <options> filter entry
```

有很多可用的 LDAP 工具，主要的有 `ldapadd`、`ldapmodify`、`ldapsearch` 和 `ldapdelete`，如上所述，这些工具都可以使用上面的通用选项。对于可用选项的详细信息，请参阅这些工具的 `man` 页面。

创建了 LDIF 文件后，在 LDAP 服务器中添加用户就是非常简单的事情了。下面来看一个完整的 LDIF 文件。前面提到过，需要在这个文件的顶部添加 `dc` 条目，即 DIT 的顶级项。

```
$ sudo cat users.ldif
dn: dc=example,dc=com
objectclass: dcObject
dc: example

dn: ou=People,dc=example,dc=com
objectclass: organizationalUnit
ou: People
dn: uid=jsmith,ou=People,dc=example,dc=com
objectclass: top
objectclass: person
objectclass: posixAccount
objectclass: exampleClient
cn: Jane Smith
sn: Smith
uid: jsmith
uidNumber: 1000
gidNumber: 1000
exampleActive: TRUE
homeDirectory: /home/jsmith
userPassword: {SHA}IOq+XWSw4hZ5boNPUtYf0LcDMvw=

dn: uid=ataylor,ou=People,dc=example,dc=com
objectclass: top
objectclass: person
objectclass: posixAccount
```



```

objectclass: exampleClient
cn: Angela Taylor
sn: Taylor
uid: ataylor
uidNumber: 1002
gidNumber: 1000
exampleActive: TRUE
homeDirectory: /home/ataylor
userPassword: {SHA}IOq+XWSw4hZ5boNPUtYf0LcDMvw=

dn: ou=meta,dc=example,dc=com
objectclass: organizationalUnit
objectclass: top
ou: meta

dn: cn=Webadmin,ou=meta,dc=example,dc=com
objectClass: organizationalRole
objectclass: simpleSecurityObject
userPassword: {SHA}IOq+XWSw4hZ5boNPUtYf0LcDMvw=

dn: ou=Groups,dc=example,dc=com
objectclass: top
objectclass: organizationalUnit
ou: Groups

dn: cn=staff,ou=Groups,dc=example,dc=com
objectclass: top
objectclass: posixGroup
gidNumber: 1000
cn: staff

dn: cn=admins,ou=Groups,dc=example,dc=com
objectclass: top
objectclass: groupOfNames
cn: admins
member: uid=ataylor,ou=People,dc=example,dc=com

dn: ou=Hosts,dc=example,dc=com
objectclass: top
objectclass: organizationalUnit
ou: Hosts

```

现在可以使用 `users.ldif` 文件添加用户了。`ldapadd` 工具的使用非常灵活，并且带有很多选项。此处使用格式如下所示。

```
$ ldapadd -xWv -D cn=admin,dc=example,dc=com -h ldap.example.com -Z -f users.ldif
```

`ldapadd` 命令可以使用 SASL 认证方式或其他简单的方式。`-x` 表示 `ldapadd` 使用简单的认证方式；`-w` 告诉 `ladpadd` 给出密码提示；`-v` 表示要详细列出命令的执行过程。使用 `-D` 选项时，要给出绑定的用户名。上例中使用用户“`cn = admin, dc = example, dc = com`”，如果大家记得，这是之前在 `slapd.conf` 中加入的根 DN。`-h` 选项指定主机名称为 `ldap.example.com`。`-Z` 告诉 `ldapadd` 命令使用 STARTTLS，或与 LDAP 服务器建立一个 TLS 连接，但是如果已经在 `/etc/ldap/ldap.co nf` 或 `/etc/openldap/ldap.conf` 文件中设置了 TLS，命令会失败。最后，`-f` 指定要在其中加入用户的文件“`users.ldif`”。

此处使用的选项对其他所有的 LDAP 工具都是一样的，更多细节可以参考 `man` 页面。运

行此命令后，会有如下输出信息。

```
jsmith@ldap:/etc/ldap$ ldapadd -xWv -D cn=admin,dc=example,dc=com ➡
-h ldap.example.com -Z -f users.ldif
ldap_initialize( ldap://ldap.example.com )
Enter LDAP Password:
add objectclass:
    top
    person
    exampleClient
    posixAccount
add cn:
    Jane Smith
add sn:
    Smith
add uid:
    jsmith
add uidNumber:
    1000
add gidNumber:
    1000
add exampleActive:
    TRUE
add homeDirectory:
    /home/jsmith
add userPassword:
    {SHA}IOq+XWSw4hZ5boNPUtYf0LcDMvw=
adding new entry "uid=jsmith,ou=People,dc=example,dc=com"
modify complete
```

若命令成功运行，最后会显示“**modify complete**”信息。如果运行中出现任何错误，可能会显示一个出错信息。

16.6.3 搜索 LDAP 树

到目前为止 LDAP 数据库中已经有了一些条目，可以试着对它进行搜索以确定能否返回有用的信息。看看可以用来搜索 LDAP 目录的方式。

```
$ ldapsearch -xvW -h ldap.example.com -Z \
-D uid=ataylor,ou=People,dc=example,dc=com \
-b ou=People,dc=example,dc=com -s sub \
'(&(&(objectclass=person)(uid=jsmith))(exampleActive=TRUE))' cn
```

搜索时使用的参数与在 `ldapadd` 命令中使用的很相似。首先，`-xvW` 确定使用简单的认证方式，显示详细的运行信息，同时希望用户提供运行密码。`-h` 指定连接的主机，`-Z` 说明使用 TLS 建立连接（`-ZZ` 表示确认 TLS 连接成功以后再继续运行）。Angela Taylor 是“`cn = admin, ou = Groups, dc = example, dc = com`”中添加的一个用户，记住，已经在访问控制列表中设定“`ou = People, dc = example, dc = com`”下所有条目的访问属性为可写。Jane Smith 是刚添加到 LDAP 目录中的用户，下面运行一个搜索来获取它的详细信息。

在 `ldapsearch` 命令中，通过使用过滤器启用索引来减少搜索响应时间。我们知道所有的

用户条目都有一个 `person` 类。如前所述, `slapd.conf` 文件中的所有对象类都进行了索引, 所以选择一个要搜索的内容所在条目的索引类将加快搜索的速度。由于 `Uid` 属性也索引了, 因此希望对所搜索条目的 `uid` 也进行过滤。搜索过滤器构造为 “`(&(&(objectclass = person)(uid=jsmith))(exampleActive = TRUE))`”, 表示对 `person` 对象类、“`uid = person`”以及“`exampleActive = TRUE`”所表示的活动账户进行索引。`&`操作符表示对操作符后面的所有关键词进行搜索, 也可以用 `|`操作符表示对操作符后面的其中一个关键词进行搜索。

使用 “`-b ou = People, dc = example, dc = com`” 表示本次搜索的起始点, 本次搜索的范围是 “`-s sub`” 及其下所有内容, 最后指定要搜索 `cn` 或通用名 `Jane`。搜索的结果会显示如下内容。

```
ldap_initialize( ldap://ldap.example.com )
filter: (&(&(objectclass=person)(uid=jsmith))(exampleActive=TRUE))requesting: cn
# extended LDIF
#
# LDAPv3
# base <ou=People,dc=example,dc=com> with scope subtree
# filter: (&(&(objectclass=person)(uid=jsmith))(exampleActive=TRUE))
# requesting: cn
#
# jsmith, People, example.com
dn: uid=jsmith,ou=People,dc=example,dc=com
cn: Jane Smith

# search result
search: 3
result: 0 Success

# numResponses: 2
# numEntries: 1
```

可以看出, 结果返回了要查找的 DN 及其条目的通用名。下面来看如何删除条目。

16.6.4 从 LDAP 中删除条目

需要经常进行的另外一个操作是从 LDAP 目录中删除条目。删除条目时, 使用 `ldapdelete` 命令。再次说明, 此命令所用的参数与 `ldapadd` 和 `ldapsearch` 一样。若一次删除多个条目, 可以把要删除的条目输入到一个文本文件中, 也可以手动一条一条地删除。假设想删除在一个新的 `users.ldif` 文件中的以下条目, 如下所示。

```
uid=jbob,ou=People,dc=example,dc=com
uid=tbird,ou=People,dc=example,dc=com
```

可以把这两个条目添加到一个称为 `deluser.ldif` 的文件中, 然后使用 `-f` 参数运行 `ldapdelete` 命令, 如下所示。

```
ldapdelete -xvW -D uid=ataylor,ou=People,dc=example,dc=com \
-h ldap.example.com -Z -f deluser.ldif
ldap_initialize( ldap://ldap.example.com )
deleting entry "uid=jbob,ou=People,dc=example,dc=com"
deleting entry "uid=tbird,ou=People,dc=example,dc=com"
```

命令执行的结果是这两个条目被删除, 不再存在于目录中。

■注: OpenLDAP 是大小写不敏感的,那就意味着“uid=jsmith,ou=People,dc=example,dc=com”和“uid=jSmith,ou=people,dc=example,dc=com”被视为同一个条目。如果尝试添加两个 Jane Smiths,一个用小写的 s,一个用大写的 S,将会返回一个重复错误。

16.6.5 密码策略覆盖

设置密码策略覆盖可以允许进行如设置密码过期期限或更改历史记录设置等高级操作。如前所述,覆盖为 OpenLDAP 服务器提供了附加的功能。这里,把密码过期期限设置为 7776000 (90 天的秒数),把密码历史设为 3,就是要保存用户前三次输入的密码以保证用户不能持续使用同一个密码。密码最小字符数为 8。

回忆一下为引入密码策略而在 slapd.conf 文件中进行的如下设定。

```
moduleload ppolicy.la
overlay ppolicy
ppolicy_default "cn=default,ou=Policies,dc=example,dc=com"
ppolicy_use_lockout
```

现在定义密码策略。要定义密码策略,还要先在 LDAP 服务器中添加如下 LDIF 设置。

```
dn: cn=default,ou=policies,dc=example,dc=com
objectClass: top
objectClass: device
objectClass: pwdPolicy
cn: default
pwdAttribute: 2.5.4.35
pwdMaxAge: 7776000
pwdExpireWarning: 6912000
pwdInHistory: 3
pwdCheckQuality: 1
pwdMinLength: 8
pwdMaxFailure: 4
pwdLockout: TRUE
pwdLockoutDuration: 1920
pwdGraceAuthNLimit: 0
pwdFailureCountInterval: 0
pwdMustChange: TRUE
pwdAllowUserChange: TRUE
pwdSafeModify: FALSE
```

■警告: 有人警告过,有关密码策略的文件中有一个严重的错误。文件中说要为 pwdAttribute 使用 userPassword 属性,其实这是错误的,解决这个问题十分令人沮丧的。应该为 objectIdentifierMatch 规则使用数字 ID 来代替。

这样就在 LDAP 服务器中添加了密码策略,而且此密码策略也进行了诸如密码过期 (90 天的秒数) 及密码历史记录 (3 条) 等基本设置。现在,覆盖会使所有的密码账户应用密码策略。

■注：完整的属性信息可以在 man 页和以下网址找到，见“<http://linux.die.net/man/5/slapo-ppolicy>”。

16.6.6 测试访问控制列表

可能会因为访问控制列表运行不正确而时不时地碰到访问权限问题，因此可以使用一个叫做 `slapacl` 的工具对 ACL（访问控制列表）进行测试。此工具可以对授权访问的 DN 进行属性和对象类访问能力测试，例如用于认证期间的 Web 服务绑定的用户 DN “`cn = Webadmin, ou = meta, dc = example, dc = com`”，如果要确认此用户具有对 Angela Taylor 用户 `userPassword` 属性的 `auth` 访问权限，可以运行如下命令。

```
sudo slapacl -f /etc/ldap/slapd.conf \
  -b uid=ataylor,ou=People,dc=example,dc=com \
  -D cn=Webadmin,ou=meta,dc=example,dc=com \
  -v userPassword/auth
```

`slapacl` 命令需要 `sudo` 系统权限。在 Ubuntu 主机上需要使用“-f /etc/ldap/slapd.conf”指定要测试的 `slapd.conf` 文件；而在 Red Hat 主机上是“-f/etc/openldap/slapd.conf”。“-b uid = ataylor, ou = People, dc = example, dc = com”指定要在其上进行测试的 DN。“-D cn = Webadmin, ou = meta, dc = example, dc = com”指定要确认是否是对 DN “uid = ataylor, ou = People, dc = example, dc = com”具有 `auth` 访问权限的 DN。还指定了想要测试的属性和认证等级，此处是要验证对 `userPassword` 属性的 `auth` 访问权限。大家知道，至少需要对 DN “webadmin, ou = meta, dc = example, dc = com”有 `auth` 权限才能使用 `userPassword` 进行认证。若运行成功，将会输出以下信息。

```
authcDN: "cn=Webadmin,ou=meta,dc=example,dc=com"
auth access to userPassword: ALLOWED
```

这就证明了访问控制列表如期望般运行。

```
access to attrs=userPassword,shadowLastChange,sambaNTPassword,sambaLMPassword
  by dn.exact="cn=Webadmin,ou=meta,dc=example,dc=com" auth
```

还要测试一下是否可以获得相同属性的写权限以验证访问控制列表有无漏洞。

```
sudo slapacl -f /etc/ldap/slapd.conf \
  -b uid=ataylor,ou=People,dc=example,dc=com \
  -D cn=Webadmin,ou=meta,dc=example,dc=com \
  -v userPassword/write
authcDN: "cn=Webadmin,ou=meta,dc=example,dc=com"
write access to uid: DENIED
```

输出信息是所期望的，即除了 `auth` 及更低的访问权限，其他都是拒绝访问的。也可以在命令中引入其他选项以测试诸如 `peername`s 和 `ssf` 等内容。

其他可以用来弄清楚访问控制列表运行情况的方法中，有一个可能是不可信的尝试，那就是测试时在访问控制列表中包含以下内容。

```
access to * by * search
```


把此行和以下 logfile 及 loglevel 两行日志设置合在一起添加到 slapd.conf 文件中。

```
Loglevel    416
Logfile     /var/log/ldap.log
```

这样会把搜索过滤器、访问控制列表运行状况、连接管理以及配置文件运行状况等信息记入日志中。当有请求时，将会有如下输出信息。

```
slapd[29981]: conn=0 op=2 SRCH base="ou=people,dc=example,dc=com" scope=2 deref=3 ➡
filter="(&(objectClass=*)(uid=ataylor))"
slapd[29981]: conn=0 op=2 SRCH attr=uid
slapd[29981]: => access_allowed: search access to "ou=People,dc=example,dc=com" ➡
"entry" requested
slapd[29981]: => dn: [2] ou=meta,dc=example,dc=com
slapd[29981]: => dn: [3] ou=people,dc=example,dc=com
slapd[29981]: => acl_get: [3] matched
slapd[29981]: => acl_get: [3] attr entry
slapd[29981]: => acl_mask: access to entry "ou=People,dc=example,dc=com", attr ➡
"entry" requested
slapd[29981]: => acl_mask: to all values by "cn=webadmin,ou=meta, ➡
dc=example,dc=com", (=0)
slapd[29981]: <= check a_dn_pat: cn=webadmin,ou=meta,dc=example,dc=com
slapd[29981]: <= acl_mask: [1] applying read(=rscxd) (stop)
slapd[29981]: <= acl_mask: [1] mask: read(=rscxd)
slapd[29981]: => slap_access_allowed: search access granted by read(=rscxd)
slapd[29981]: => access_allowed: search access granted by read(=rscxd)
```

第一行是访问请求的搜索字符串“SRCH base = “ou = people,dc=example,dc=com” scope=2 deref=3 filter=“(objectClass=*)(uid=ataylor))””，输出信息也显示了提出访问请求的用户“cn=webadmin,ou=meta,dc=example,dc=com”。从输出信息中可以看到对“ou = People, dc = example, dc = com”的搜索访问请求以及最后请求获准的过程，请求获准是因为 webadmin 用户对“ou = People, dc = example, dc = com”下的所有内容都有只读访问权限。

凭借日志、slapacl 和 ldapsearch 工具组合以及非常有用的 OpenLDAP 邮件列表，可以实现非常复杂的访问控制列表。接下来看一下可以用来管理 LDAPsearch 服务器的其他工具，包括刚刚提到过的 ldapsearch 工具。

16.6.7 备份 LDAP 目录

基于文本的文件对于创建和恢复 LDAP 服务器都是非常重要的。一旦目录部署好，建议编写一个脚本，用来经常将 LDAP 数据输出到文本文件并保存。第 13 章将会介绍 Bacula 备份服务器以及如何使用 Client Run Before Job 和 Client RunAfter Job 选项备份 MySQL 数据库。如列表 16-2 所示，也可以在 LDAP 数据库上进行类似的操作。

列表 16-2 对 LDIF 执行 slapcat 操作并转存

```
#!/bin/bash

case $1 in
  start)
    slapcat -b dc=example,dc=com -l /var/lib/ldap/backup.ldif
```



```

        if [ $? -eq 0 ] ; then
            echo "backup successful"
        else
            echo "backup failed"
            exit 1;
        fi
    ;;

stop)
    if [ -e /var/lib/ldap/backup.ldif ] ; then
        rm -f /var/lib/ldap/backup.ldif
        if [ $? -eq 0 ] ; then
            echo "removal of file successful"
        else
            echo "failed to remove file"
            exit 1;
        fi
    fi

fi
;;
esac
exit 0

```

要获得一个完整的备份，需要在运行列表 16-2 中的命令之前停止 OpenLDAP 目录服务器，虽然不能总是停止服务器，但是热备份总比不备份要好得多。

首先使用 Client Run Before Job 脚本，通过 `slapcat` 命令将 LDAP 数据库备份到磁盘的一个文件中。然后使用 Bacula 将其备份，Bacula 可以通过运行 Client Run After Job 脚本（如列表 16-3 所示）将其删除。

列表 16-3 Bacula 备份服务的 Job 定义

```

Job {
    Name = ldap.example.com
    Client = ldap-fd
    Enabled = yes
    JobDefs = "DefaultLinux"
    Client Run Before Job = "/usr/local/bin/ldap_backup start"
    Client Run After Job = "/usr/local/bin/ldap_backup stop"
}

```

这一切操作都建立在 LDAP 备份脚本已经安装在 `ldap.example.com` 主机中的 `/usr/local/bin` 路径下的基础之上。

接着，恢复 LDAP 数据库就变成了一个使用以下参数在主机上运行 `slapadd` 命令的简单问题（运行此命令时应该关闭 OpenLDAP）。

```
$ slapadd -b dc=example,dc=com -f /etc/ldap/slapd.conf -l restored.ldif.backup.file
```

这样就把 LDAP 数据库恢复到了最后一次执行备份时的样子。因为 LDAP 没有日志提前写功能，不能像一个功能齐全的事务型关系数据库一样将最近的更新内容恢复到 LDAP 目录服务器，因此，经常地对数据库进行备份变得非常重要，建议每天备份一次。

使用文本文件管理目录服务器可能会变得非常烦人。幸运的是，如果用户喜欢使用基于 Web 的 GUI 解决所有琐碎事情，就有另外的解决办法了，下面将介绍这个工具。

16.6.8 LDAP 账户管理：基于 Web 的 GUI

有很多工具可以用来管理 LDAP 目录。本书决定只集中讨论其中的一个，即 LDAP 账户管理工具（LAM）。这是一个基于 Web 的 GUI，它可以消除很多升级文本文件带来的管理痛苦。LAM 有两个版本，一个免费版本和一个需要交费使用的企业版本。如果大家发现不喜欢使用此工具，那可以尝试一下由以下链接处提供的其他可用工具。

- *Luma*: <http://luma.sourceforge.net/>
- *GQ*: <http://sourceforge.net/projects/gqclient/>
- *phpldapAdmin*: <http://phpldapadmin.sourceforge.net/>

选择介绍 LAM 是因为它不仅可以用来管理 LDAP，还可以管理用户账号。它允许基于易用的用户模板创建用户。它还具有足够的灵活性，如果选择这样做，它允许把 Samba 用户管理整合进来。

16.6.9 安装与配置

LAM 可以在 Ubuntu 的在线资料库中下载到。可运行如下命令进行安装。

```
$ sudo aptitude install php5-mhash ldap-account-manager
```

对于使用 RHEL 的 Red Hat 主机，需要从 SourceForge 主页（<http://lam.sourceforge.net/download/index.htm>）下载 Fedora 的 RPM 安装包，然后使用“rpm -ivh”命令进行安装。

Fedora 主机上的 LAM 可以从在线 Yum 资料库中找到。Red Hat 中安装的 LAM 将把所有文件都放在 `/var/www/html/lam` 路径下，而把所有的配置文件都安装到了 `/var/www/html/lam/config` 路径下。

安装 LAM 之前，确认主机上已经安装了 5.1 版或更高版本的 PHP。而且要编辑 `php.ini` 文件，将内存限制设为如下所示的 64MB。Ubuntu 中，`php.ini` 文件可以在 `/etc/php5/apache2/php.ini` 路径中找到，在 Red Hat 主机上的路径为 `/etc/php.ini`。

```
memory_limit = 64M
```

LAM 是很容易配置的。在 Ubuntu 中，许多配置文件都被安装到了 `/etc/ldap-account-manager` 中。`/etc/ldap-account-manager/config.cfg` 是 Apache Web 服务器的一个示例配置文件，其中包含了 LAM 安装的默认用户名和密码。

```
$ sudo vi /etc/ldap-account-manager/config.cfg
# password to add/delete/rename configuration profiles
password: somepassword
```

```
# default profile, without ".conf"
default: lam
```

在 `config.cfg` 文件中可以看到用户密码为 `somepassword`（明文，所以要小心设置所有权和权限）。另外还要修改以下文件，以添加自己的 LDAP 目录的细节信息。首先，复制一个 `/var/lib/ldap-account-manager/config/lam.conf` 文件副本，然后按照黑体字显示的改变此文件。

```
$ sudo vi /var/lib/ldap-account-manager/config/lam.conf
# LDAP Account Manager configuration
```



```
# server address (e.g. ldap://localhost:389 or ldaps://localhost:636)
ServerURL: ldaps://ldap.example.com

# list of users who are allowed to use LDAP Account Manager
Admins: cn=admin,dc=example,dc=com

# password to change these preferences via Webfrontend
Passwd: somepassword

# suffix of tree view
treesuffix: dc=example,dc=com

# maximum number of rows to show in user/group/host lists
maxlistentries: 30

# default language (a line from config/language)
defaultLanguage: en_GB.utf8:UTF-8:English (Great Britain)

# Number of minutes LAM caches LDAP searches.
cachetimeout: 5

# Module settings
modules: posixAccount_minUID: 1000
modules: posixAccount_maxUID: 30000
modules: posixAccount_minMachine: 50000
modules: posixAccount_maxMachine: 60000
modules: posixGroup_minGID: 1000
modules: posixGroup_maxGID: 20000
modules: posixGroup_pwdHash: SSHA
modules: posixAccount_pwdHash: SSHA
```

在此文件的第一部分，添加了包括连接信息、树信息、Posix UID、GID 以及主机数目等 LDAP 目录详细信息。

```
# List of active account types.
activeTypes: user,group,host

types: suffix_user: ou=People,dc=example,dc=com
types: attr_user: #uid;#givenName;#sn;#uidNumber;#gidNumber
types: modules_user: person,posixAccount,shadowAccount,exampleClient

types: suffix_group: ou=Groups,dc=example,dc=com
types: attr_group: #cn;#gidNumber;#memberUID;#description
types: modules_group: posixGroup

types: suffix_host: ou=Hosts,dc=example,dc=com
types: attr_host: #cn;#description;#uidNumber;#gidNumber
types: modules_host: account,posixAccount

# Access rights for home directories
scriptRights: 750
```

此文件的最后部分，在 activeTypes 部分详细描述了希望允许的用户类型以及包含这些用户类型的 LDAP 分支，这些信息将被 LAM 管理工具用来在 LDAP 服务器中创建用户账号。

在最初的 lam.conf 文件中，还有关于管理 Samba 用户账号的参考信息。如果愿意，可以在适当的位置包含这些信息。

16.6.10 为 LAM 添加 Apache 虚拟主机

下面，将要在 Web 服务器中添加一个 Apache 虚拟主机作为 LAM 站点的主机。在第 11

章中已经讲过如何设置一个 Apache 虚拟主机。Web 服务可以运行在任何一个主机上，它不必和 LDAP 服务器运行在同一个主机上，然而在此示例中，这两个服务器却是运行在了同一个主机上。这里选定使用 IP 地址 192.168.0.1 和称为 ldap.example.com、也叫 headoffice.example.com 的 DNS 来容纳主机上的这个站点。

■注：主机 headoffice.example.com 可能已经因为安全虚拟主机（https）而超负荷运转，这时就需要使用一个非标准端口（如 8443）来运行 ldap.example.com 站点。更或者，干脆在另一台完全不同的主机上运行它。

在 Ubuntu 主机上，可能会有如下所示的配置。这个虚拟主机的主要部分是由 LAM 软件包所提供的，配置文件可以在 /etc/ldap-account-manager/apache.conf 找到。所需的配置已被安装在路径 /etc/apache2/sites-available 下，并已经添加了虚拟主机的详细信息。而 Red Hat 主机上的配置文件在 /etc/httpd/conf.d/ 路径下。

```
$ sudo vi /etc/apache2/sites-available/ldap.example.com
<VirtualHost 192.168.0.1:443>
    ServerName ldap.example.com
    DocumentRoot /var/www/sites/lam
    SSLEngine on
    SSLCertificateFile /etc/ssl/certs/ldap.pem
    SSLCertificateKeyFile /etc/ssl/certs/ldap.pem
    LogFormat "%v %l %u %t \"%r\" %>s %b" comonvhost
    CustomLog /var/log/apache2/ldap.example.com/access.log comonvhost
    ErrorLog /var/log/apache2/ldap.example.com/error.log
    LogLevel debug
Alias /lam /usr/share/ldap-account-manager
<Directory /usr/share/ldap-account-manager>
    Options +FollowSymLinks
    AllowOverride All
    Order allow,deny
    Allow from all
    DirectoryIndex index.html
</Directory>
<Directory /var/lib/ldap-account-manager/tmp>
    Options -Indexes
</Directory>
<Directory /var/lib/ldap-account-manager/sess>
    Options -Indexes
    Order allow,deny
    Deny from all
</Directory>
<Directory /var/lib/ldap-account-manager/config>
    Options -Indexes
    Order allow,deny
    Deny from all
</Directory>
<Directory /usr/share/ldap-account-manager/lib>
```



```
Options -Indexes
<Files ~ .*>
    Order allow,deny
    Deny from all
</Files>
<Files ~ fpdf.php>
    Order allow,deny
    Allow from all
</Files>
</Directory>
<Directory /usr/share/ldap-account-manager/lib/font>
    Options -Indexes
    Order allow,deny
    Deny from all
</Directory>
</VirtualHost>
```

虚拟主机被包含在“<VirtualHost> </VirtualHost>”标签中。在/etc/apache/httpd.conf 文件中，还要添加如下信息。

```
NamedVirtualHost 192.168.0.10:80
```

在 Red Hat 主机上，上面的信息应被添加到文件/etc/httpd/conf.d/httpd.conf 中。下面，启动 Apache Web 服务器并在浏览器中填入地址“http://ldap.example.com”。浏览器中显示的就是 LAM 配置工具的登录页面，如图 16-3 所示。



图 16-3 LAM 登录页面

注意在右上角的 LAM 配置链接，此链接用于对 LAM 的配置进行一般性维护。在登录页上，要求填写在文件/etc/ldap-account-manager/config.cfg 中创建的密码，也可以在这里更改通用登录设置和管理密码。

图 16-3 中所示的 admin 用户指的是在文件/var/lib/ldap-account-manager/config/lam/conf中指定的根 DN。输入根 DN 的密码后，页面将会显示前面所配置的用户列表，如图 16-4 所示。

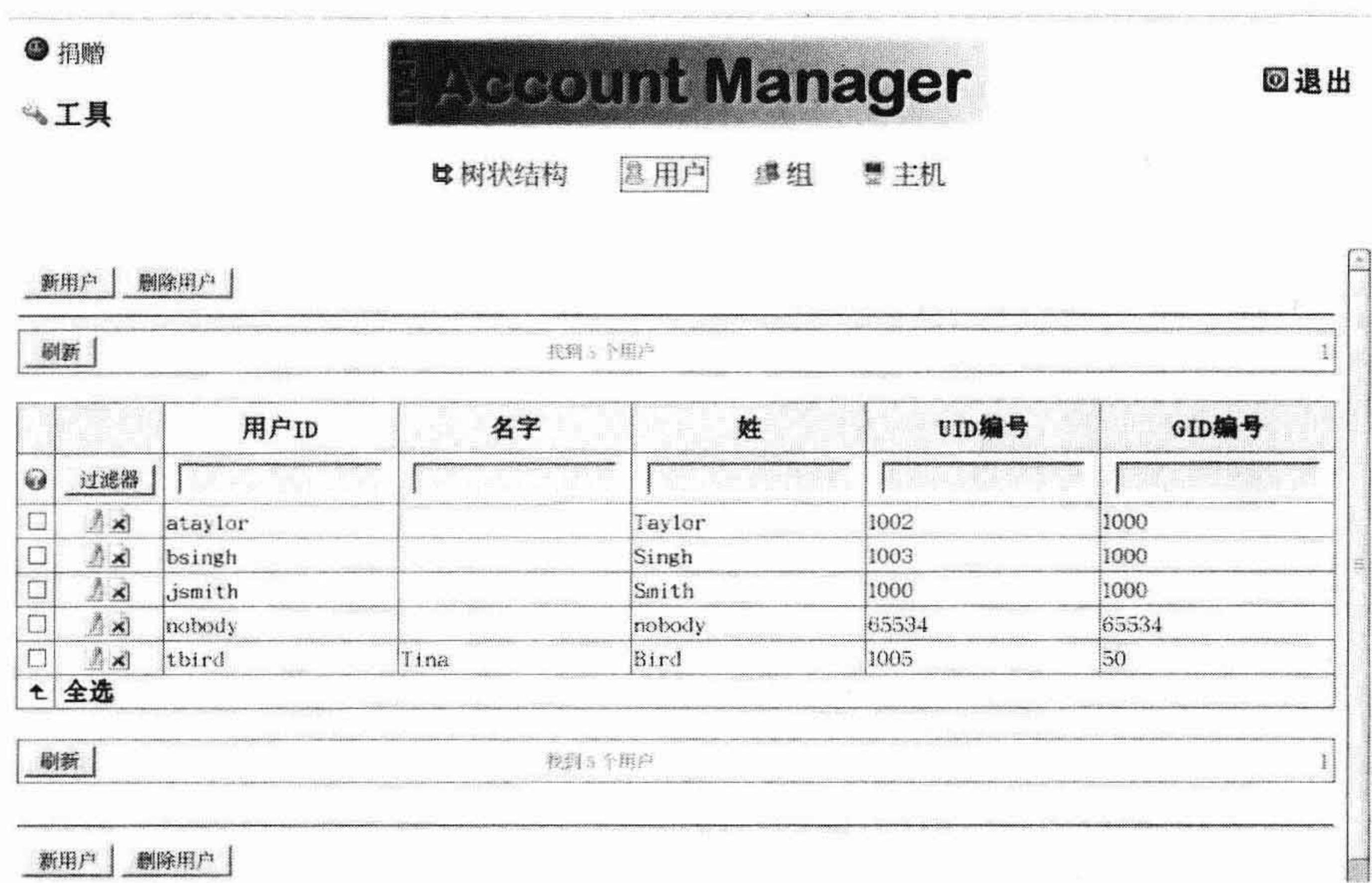


图 16-4 LAM 的 Web GUI 页面

下面就来使用标准配置参数创建一个新用户。配置参数就像是创建用户的模板，单击“新用户”按钮进入图 16-5 所示的页面。

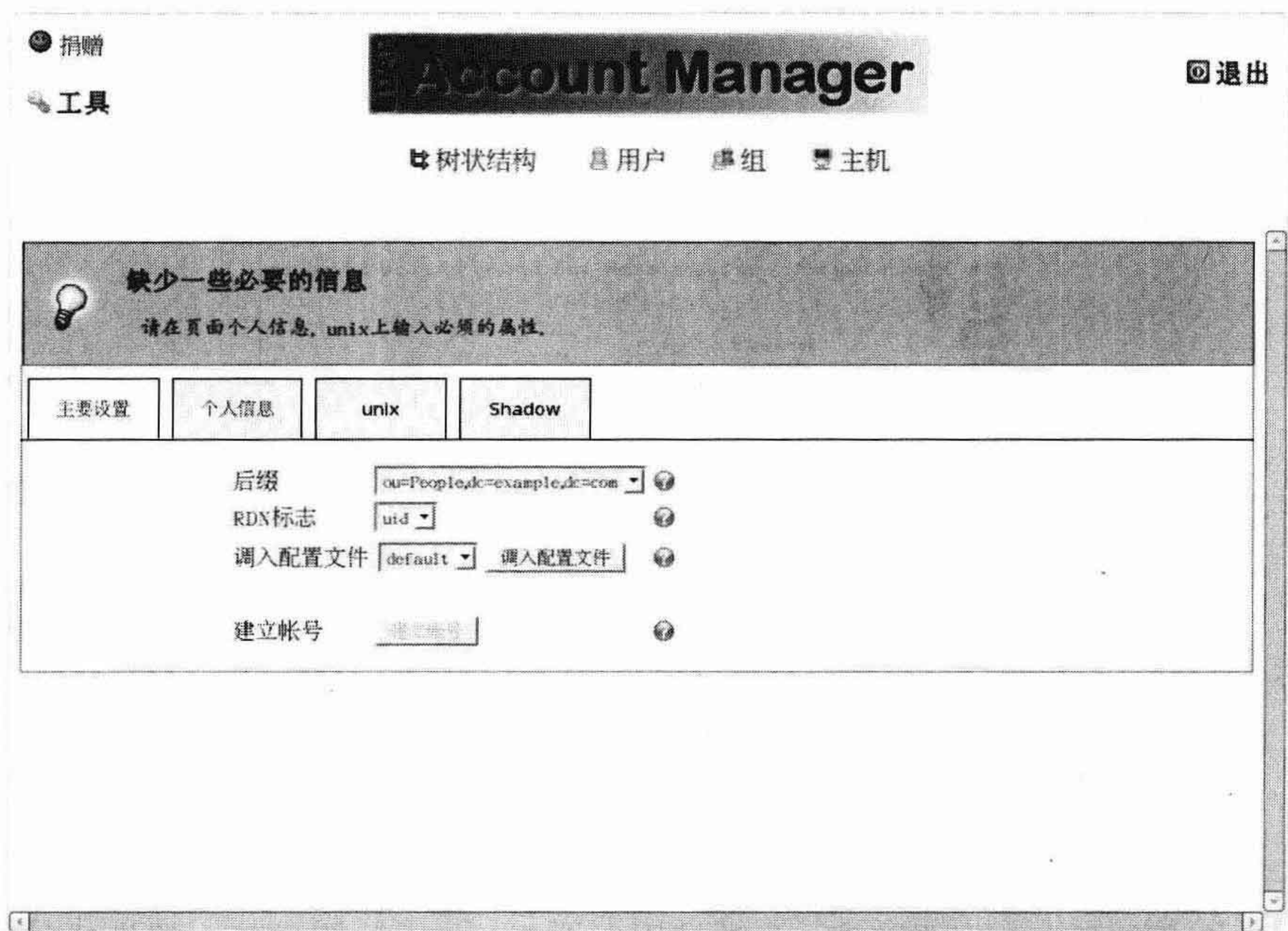


图 16-5 创建新用户

可以看到，后缀设置为“ou = People, dc = example, dc = com”，RDN 标识符设置为 uid。现在单击“个人信息”标签并填写详细信息。此处只填写姓、名以及描述项的详细信息。



图 16-6 个人详细信息

在 Unix 标签中，填写创建一个 Unix/Linux 账户所需的详细信息，如图 16-7 所示。

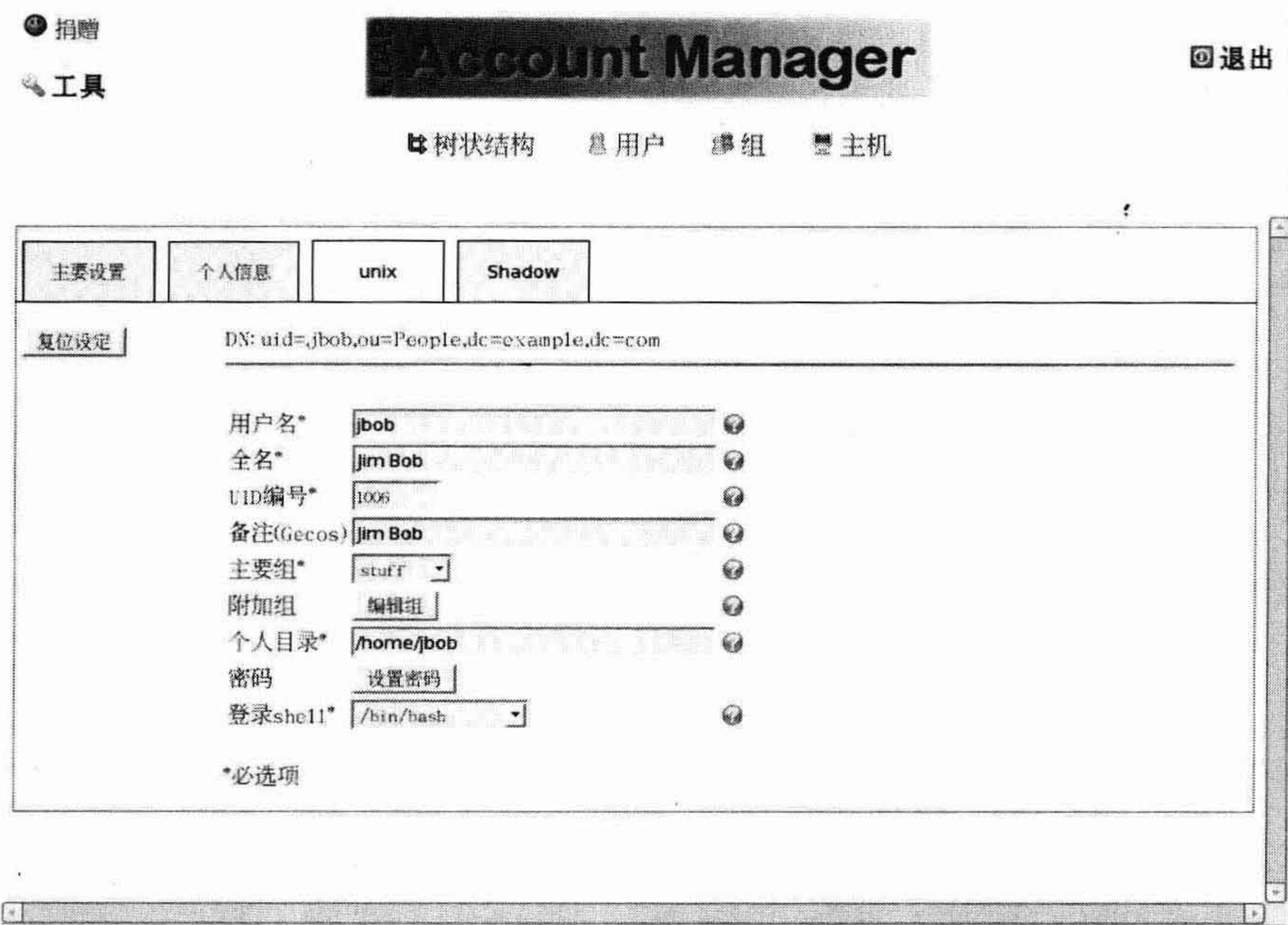


图 16-7 Unix/Linux 详细信息

在“Shadow”标签中，保留默认设置不动。默认信息如图 16-8 所示。

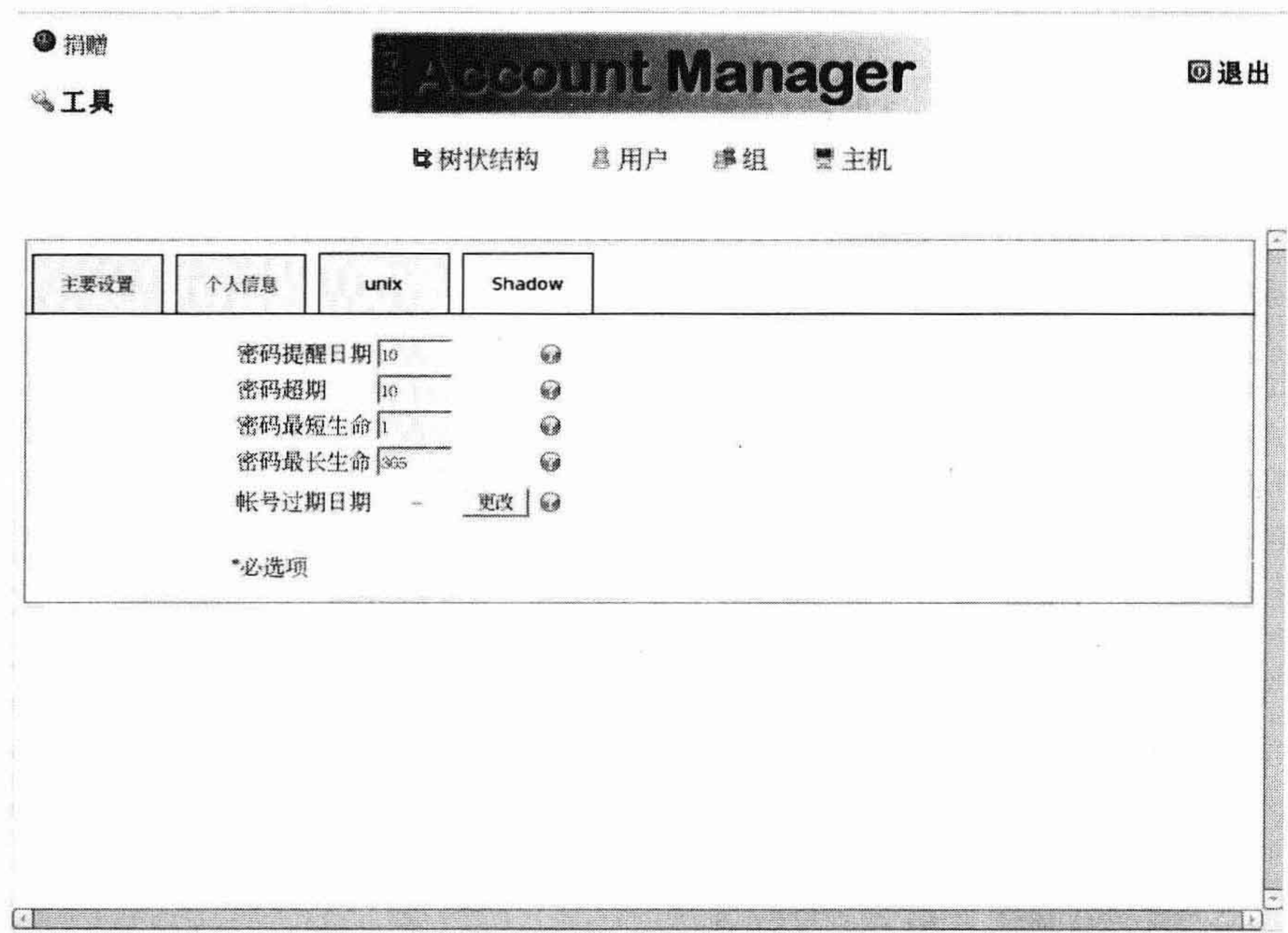


图 16-8 “Shadow” 详细信息

接下来，回到“主要设置”标签页，单击“建立账号”按钮来完成创建新用户。页面将显示一个确认框显示操作成功，如图 16-9 所示。

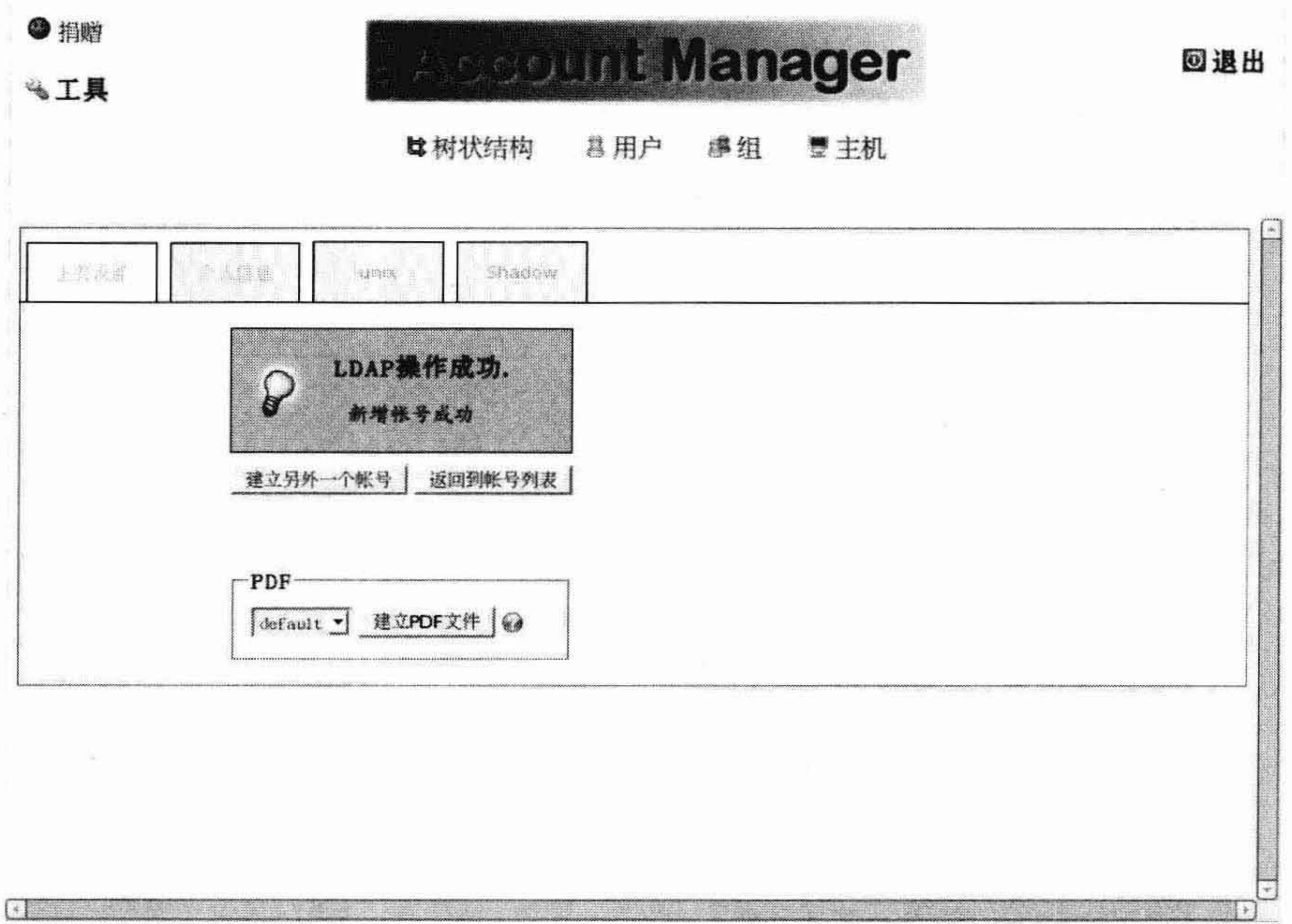


图 16-9 创建新用户完成

从图 16-10 中可以看出，新用户 jbob 创建完成。



图 16-10 LDAP 用户创建完成

还可以使用 LAM 添加、删除用户组及 LDAP 目录中的主机条目，这些内容留给读者自己进行进一步探索。记住，LAM 不是唯一的 LDAP 管理工具。如果不喜欢这个管理 LDAP 服务的 LDAP 管理客户端工具，建议试试前面提到过的其他客户端工具。

16.7 与其他服务整合

部署 LDAP 服务器的主要目的是可以通过一个认证服务整合其他需要认证的不同服务。我们希望在尽量多的服务中使用同一个用户名和密码，这将使管理员可以更好地进行用户管理，还允许为所有服务设置通用的密码管理策略，同时也提供了更高的安全性。

服务整合的第一步是集中 Linux 认证，这样就可以让所有的 Linux 桌面和服务器共享相同的认证证书。接下来，展示如何在 Web 服务中添加 LDAP 认证机制。最后会讨论一个基于 Web 的应用程序是如何使用 LDAP 进行认证服务的。

16.7.1 单点登录：集中 Linux 认证

现在来看如何集中 Linux 主机中的所有用户账号。在几个 Linux 主机上都分别有几个用户账号很可能会造成管理的麻烦，而且密码可能会停止同步，当用户离开部门后其账号并没有删除，这些都会造成潜在的安全威胁。为了简化此种情况的用户管理，可以通过把所有 Linux 主机都指向 LDAP 服务器来集中认证服务。为了演示如何做到这一点，首先看一下安装必备软件及检查配置中使用到的文件的过程。幸运的是，可以使用系统发行版提供的认证工具来配置单点登录可以运行的必要文件。

在 Ubuntu 中，必须安装下述软件包。

```
$ sudo aptitude install libpam-ldap libnss-ldap nss-updatedb libnss-db  
ldap-auth-config libpam-cracklib
```

安装 ldap-auth-config 时，将会提出几个在客户端配置 LDAP 认证服务的问题。也可以使用以下命令。

```
$ sudo dpkg-reconfigure ldap-auth-config
```

Ldap-auth-config 工具通过配置可以使用户利用 LDAP 认证主机的设置。

将来，还会允许使用 debconf（Debian 配置管理工具）控制 LDAP 服务器的配置，如图 16-11 所示。

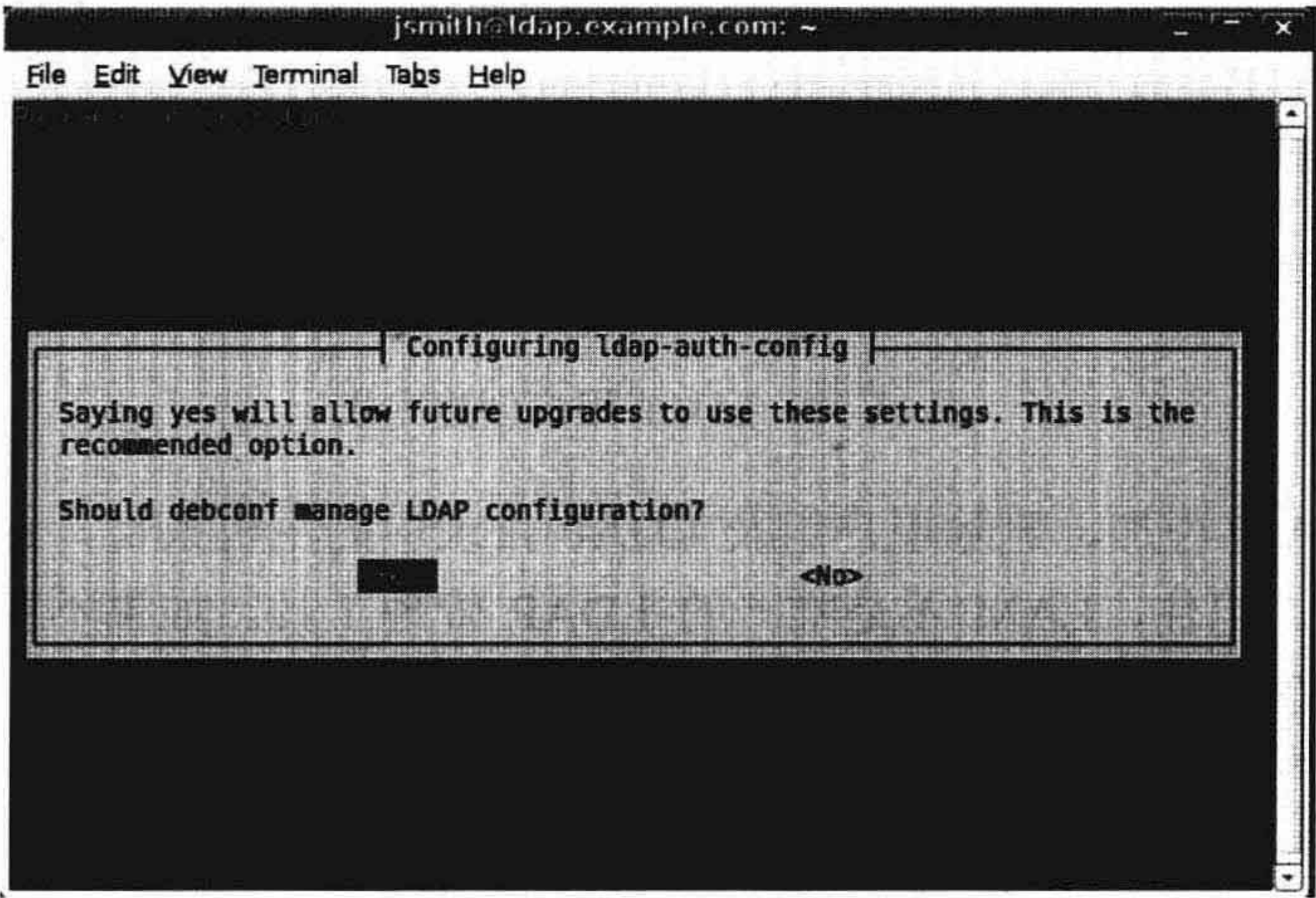


图 16-11 允许 debconf 管理 LDAP 配置

下面设置 LDAP 主机的 URI，如图 16-12 所示。

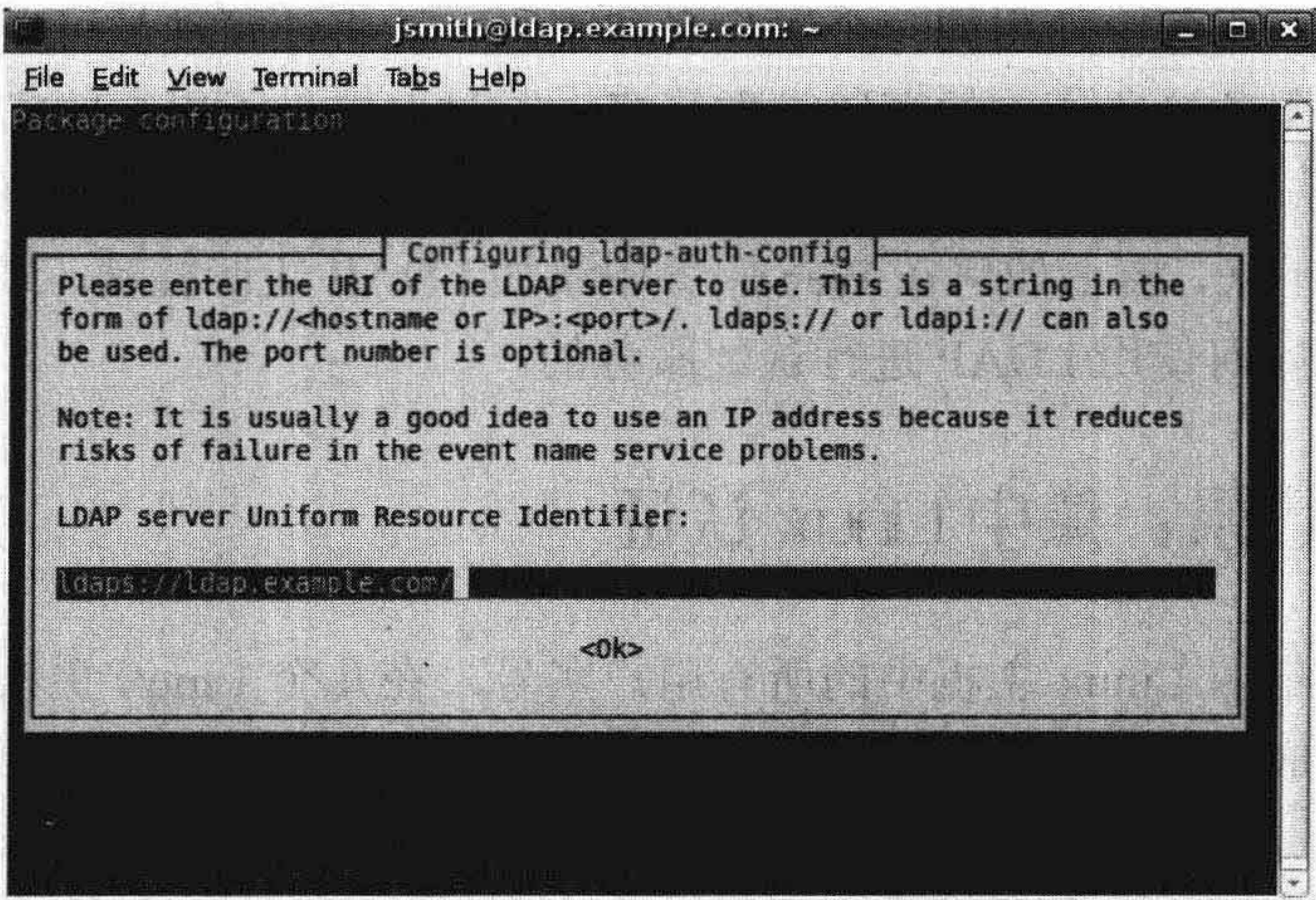


图 16-12 为 LDAP 主机设置 URI

图 16-13 所示为在 LDAP 服务中进行搜索的开始点。

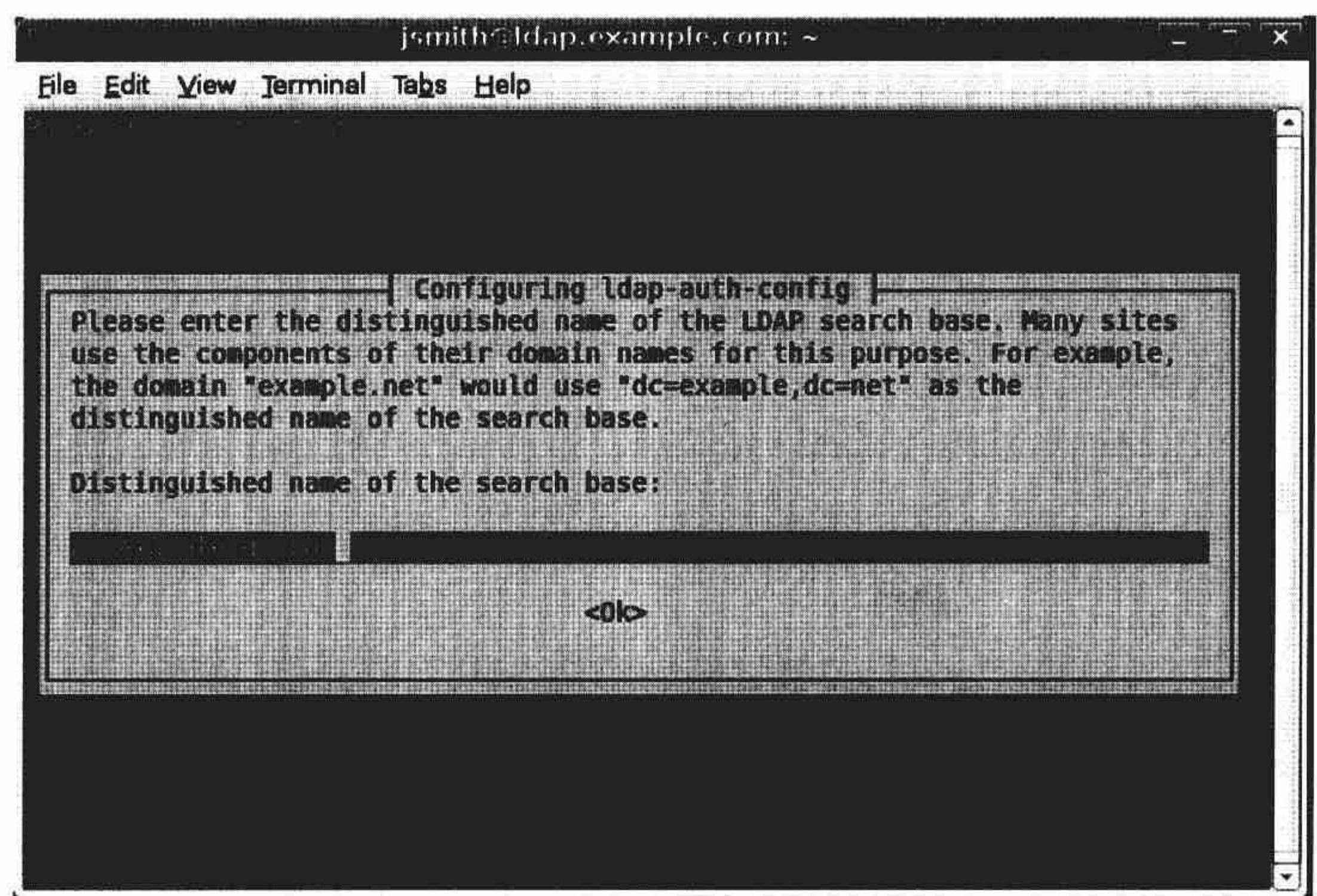


图 16-13 搜索根基

下一页，选择 LDAP 的版本为 version3，因为不推荐使用 version2（见图 16-14）。

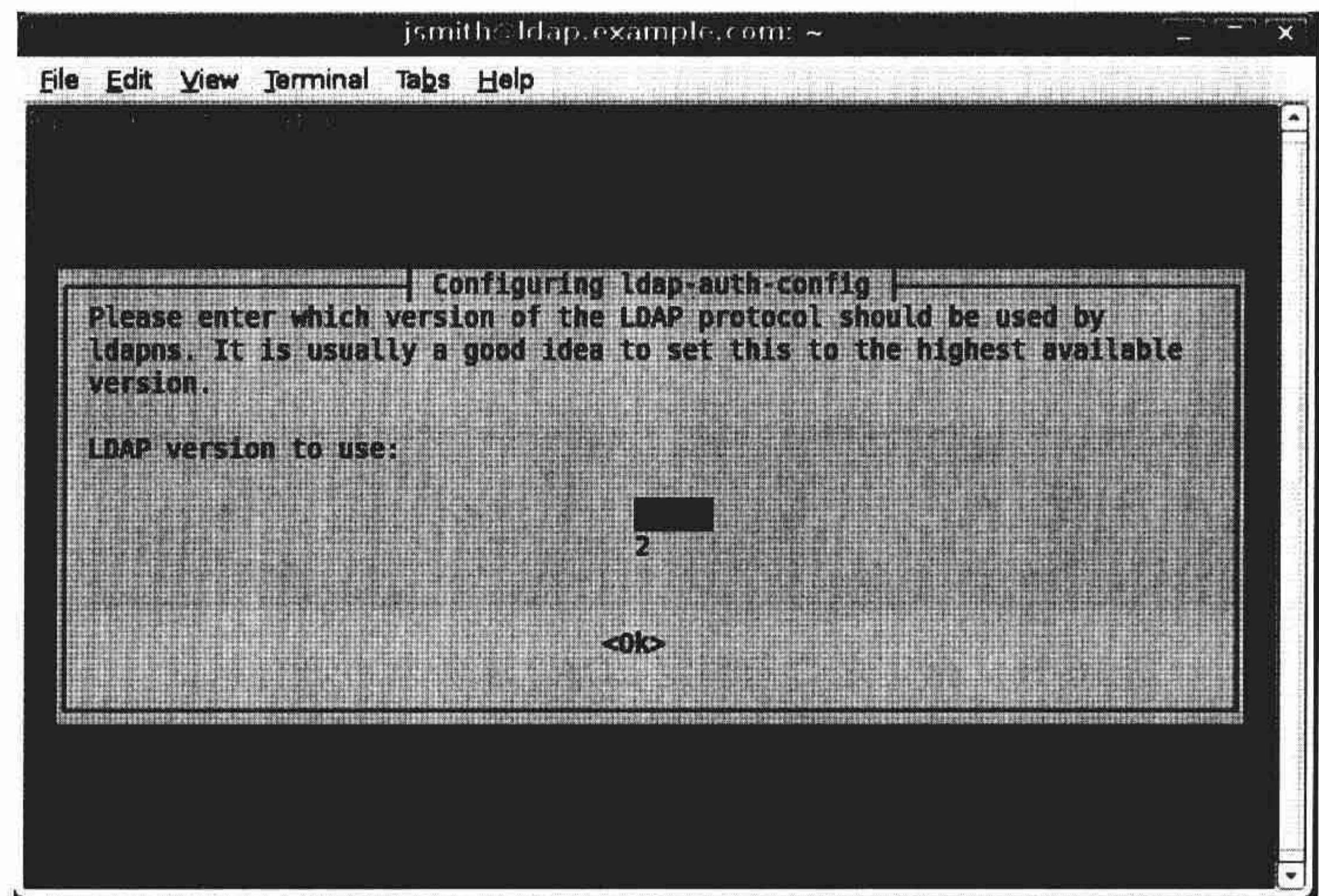


图 16-14 选择 LDAP 版本

如果愿意，可以允许主机在 LDAP 服务器上修改密码（见图 16-15）。包含 LDAP 服务器 root 用户密码的密码文件在客户机上有一个安全问题。通常这是不允许的，但是在这里允许，这只是为了能够看到下面的其他工具。

下一页，如图 16-16 所示，询问是否必须先进行 LDAP 服务认证，然后再进行个人客户端认证（使用一个单独的通用 DN 对所有客户端进行认证）。这里选择 “No”，因为此例中不要求认证。

下一页，要求制定一个可以在 LDAP 目录服务中可以更改密码的用户（见图 16-17）。这涉及前面做出过的选择，也是讨论过的安全考虑的一部分。

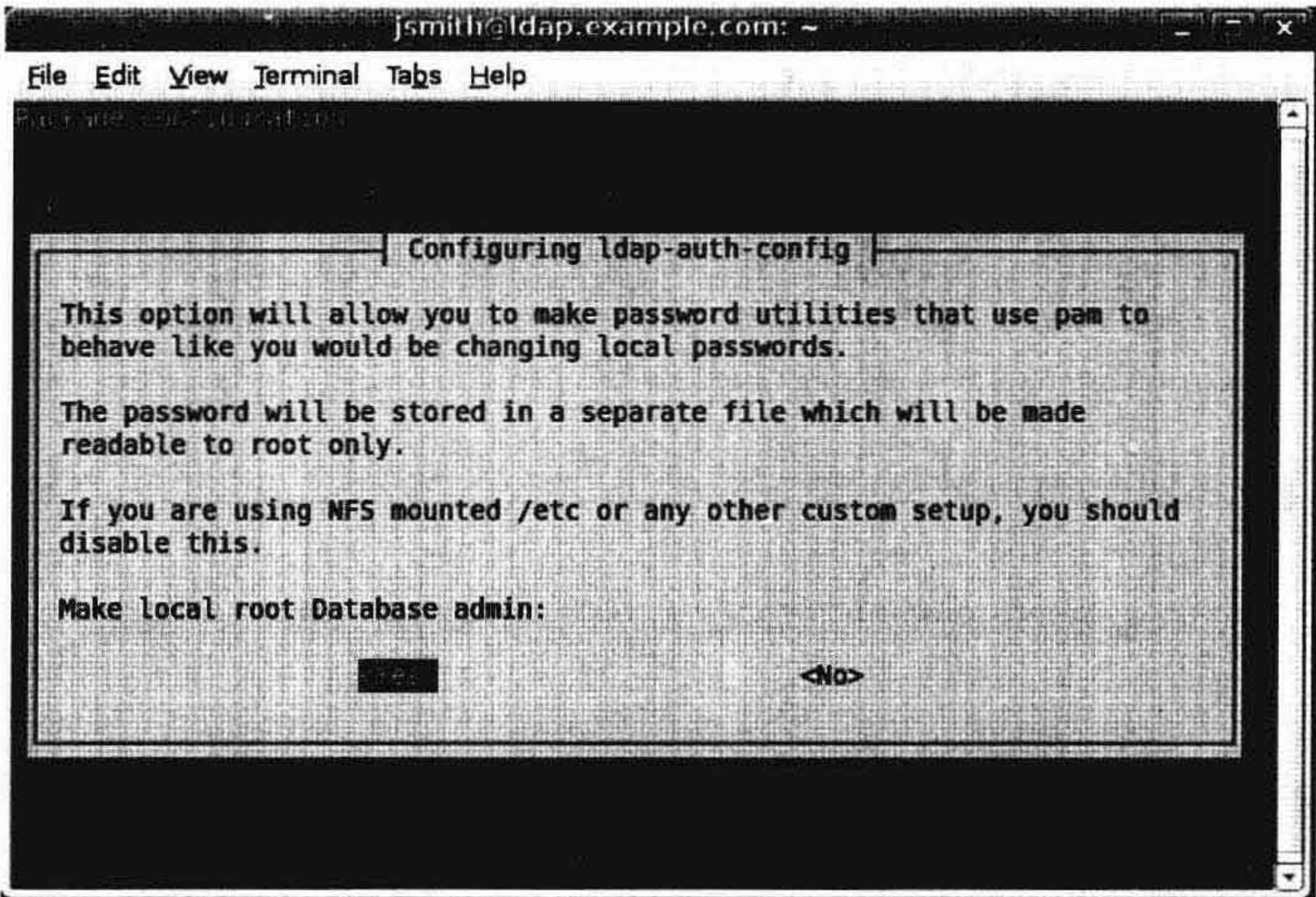


图 16-15 允许更改密码

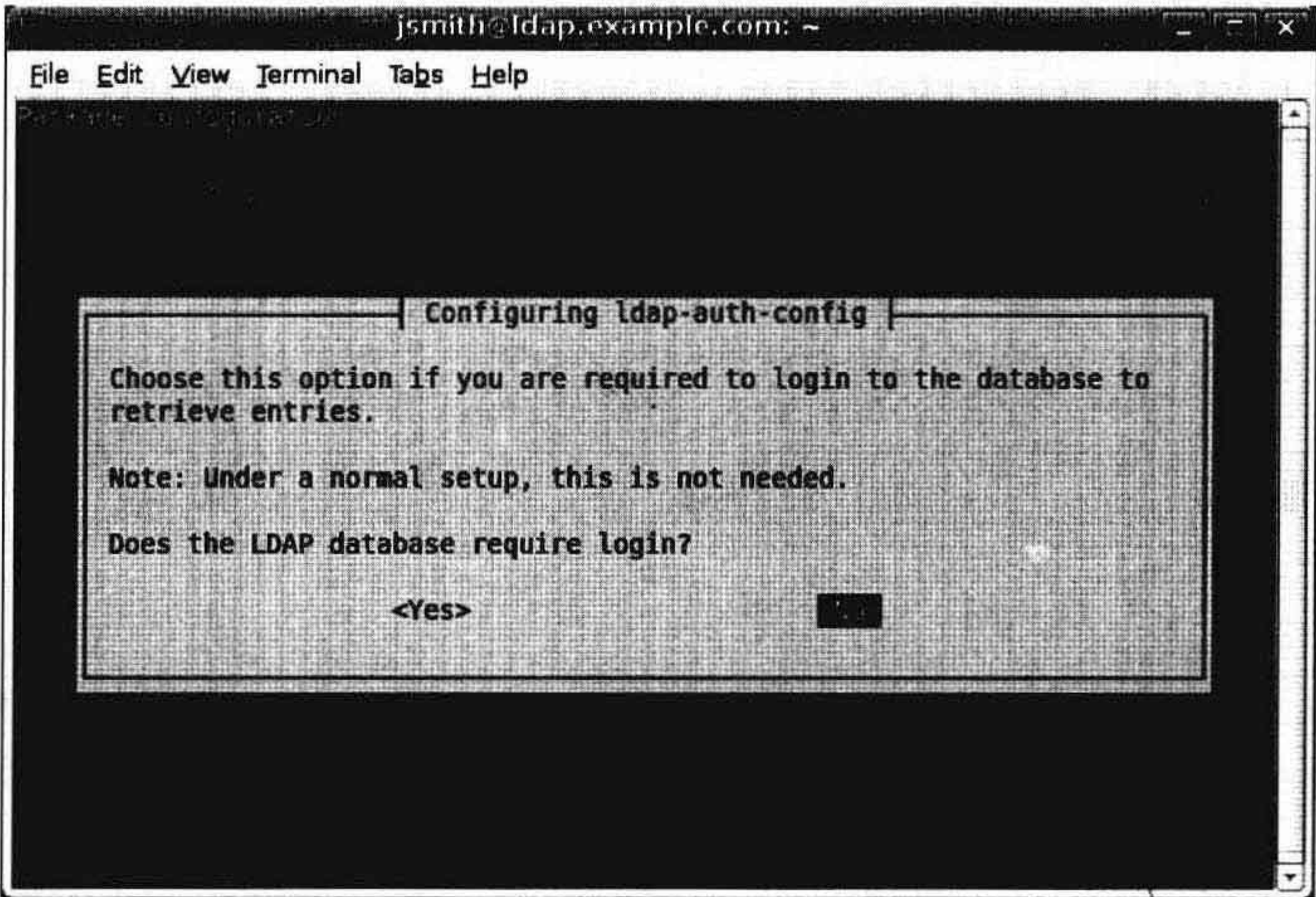


图 16-16 指定是否先绑定 LDAP 服务器再进行认证

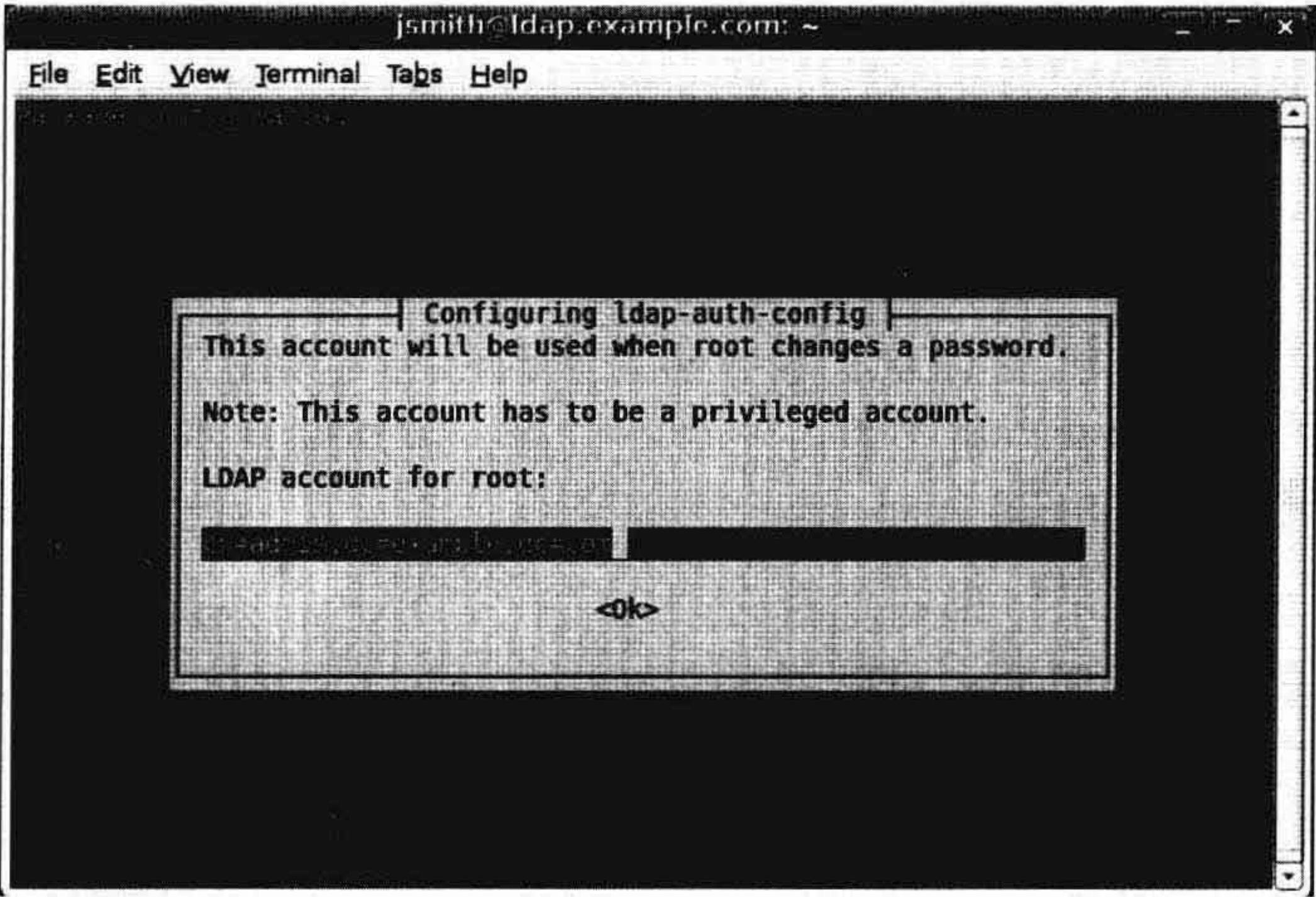


图 16-17 可以更改密码的用户

下面，需要提供 root 账户的密码，如图 16-18 所示。可以看到，页面上的信息显示文件只能被本地的 root 账户所读取，这也是出于若主机受损时的安全考虑。

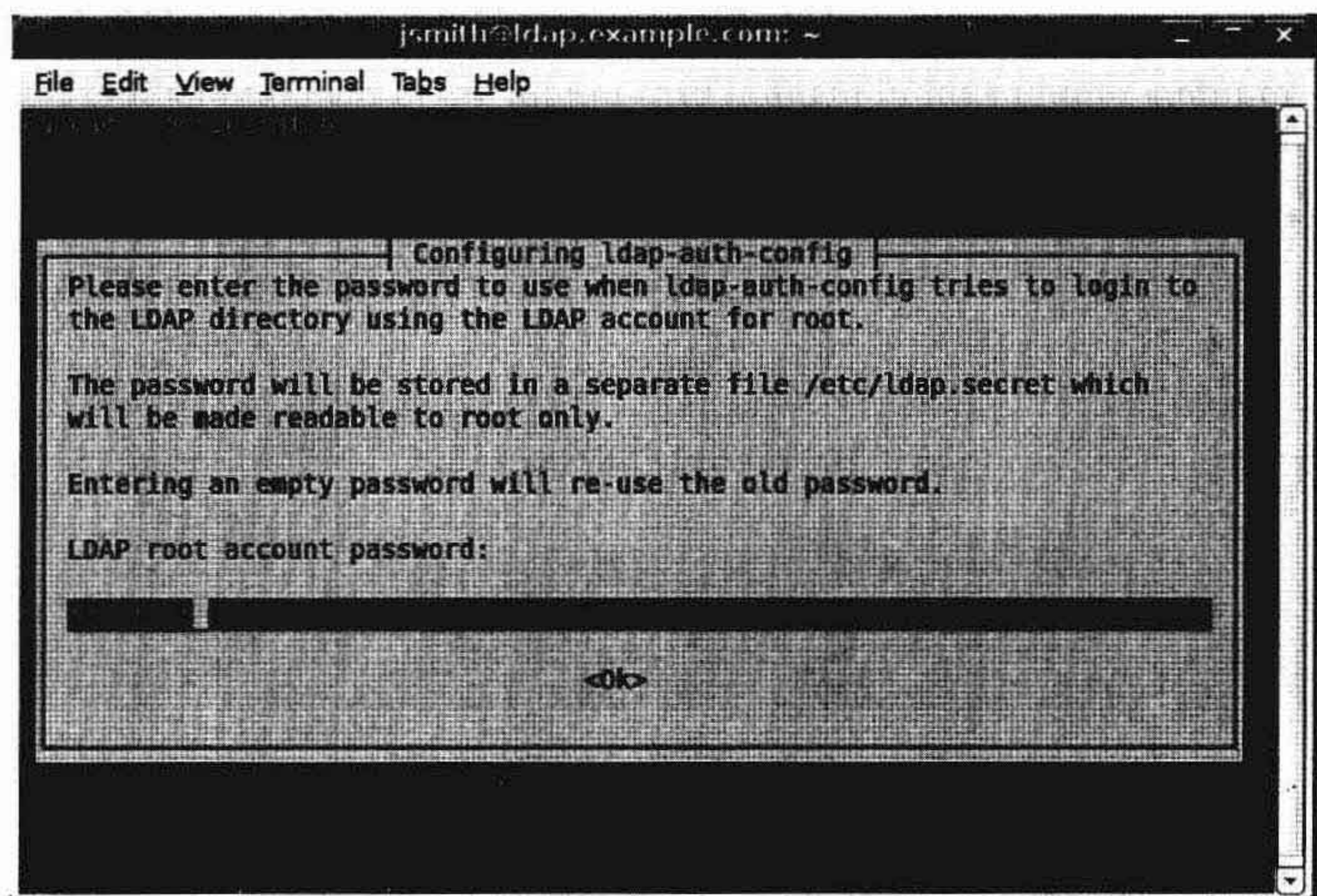


图 16-18 root 账户的密码

图 16-19 所示页面显示了密码的各种可用加密选项或加密机制，以及下一页的相关信息，如图 16-20 所示。

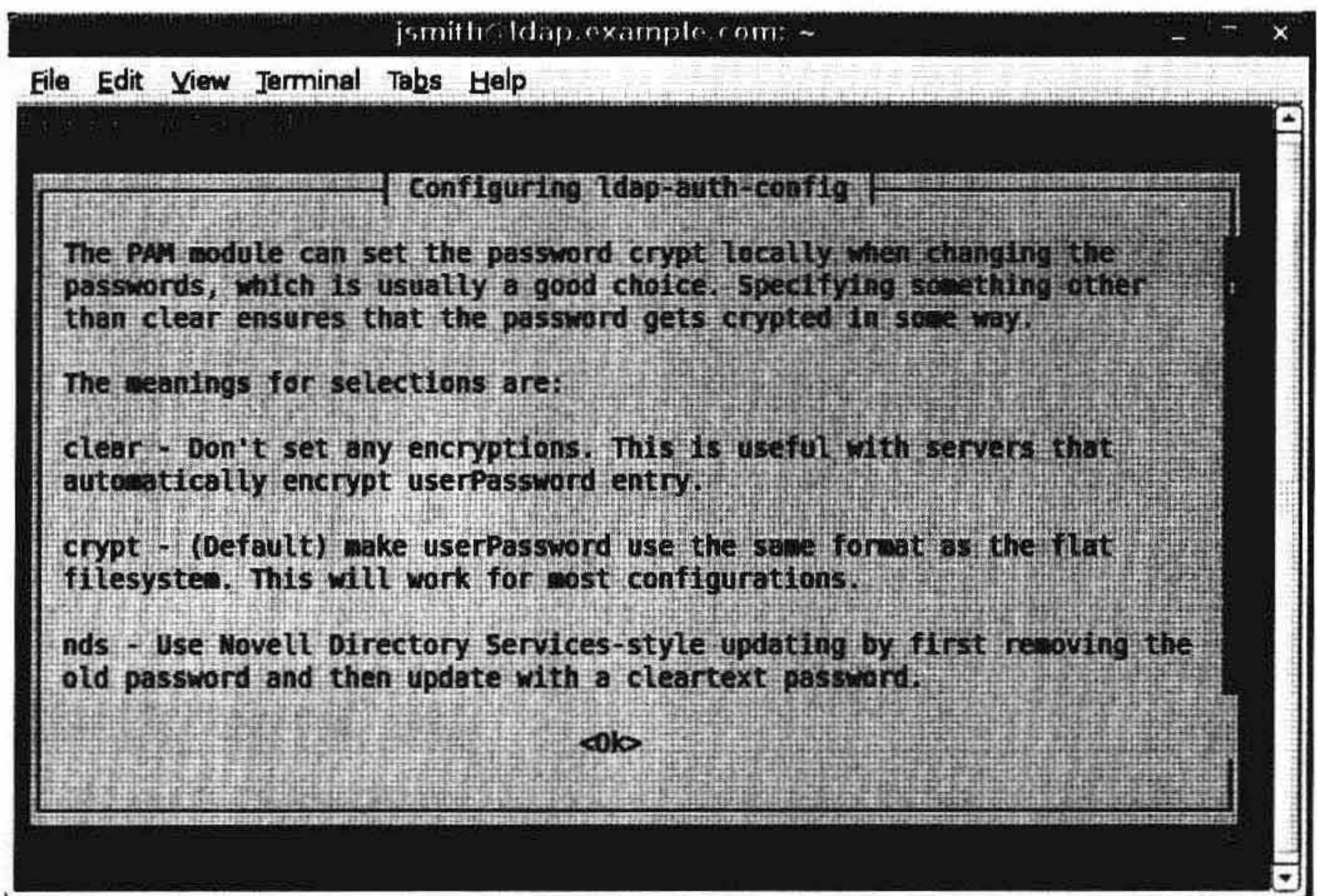


图 16-19 密码加密选项及其描述

■提示：若想对 Linux/Unix 加密有更好的理解，请参考[http://en.wikipedia.org/wiki/crypt_\(Unix\)](http://en.wikipedia.org/wiki/crypt_(Unix))。

MD5 (md5) 加密是 Linux 主机上最常用的密码加密。如果具备基础结构，其他可选项还包括 AD 服务器 (ad) 和 Novell eDirectory 服务器 (nds)。

在 Red Hat 上，需要安装如下软件包：

```
$ sudo yum install nss_ldap system-config-authentication
```




图 16-20 选择了 MD5 加密

可以使用 `system-config-authentication` 工具配置 LDAP 服务认证。在桌面主机上，可以选择“系统”>“管理”>“验证”或在命令行中运行如下命令：

```
$ sudo system-config-authentication
```

在打开的对话框中，如图 16-21 所示，使用“用户信息”（User Information）选项卡指定 LDAP 提供的主机用户信息，可以允许只从 LDAP 中取得用户信息而从其他服务获取认证信息。这里通过选中相应的复选框来允许 LDAP 支持。

接着，单击“配置 LDAP”按钮打开 LDAP 配置对话框，如图 16-22 所示。

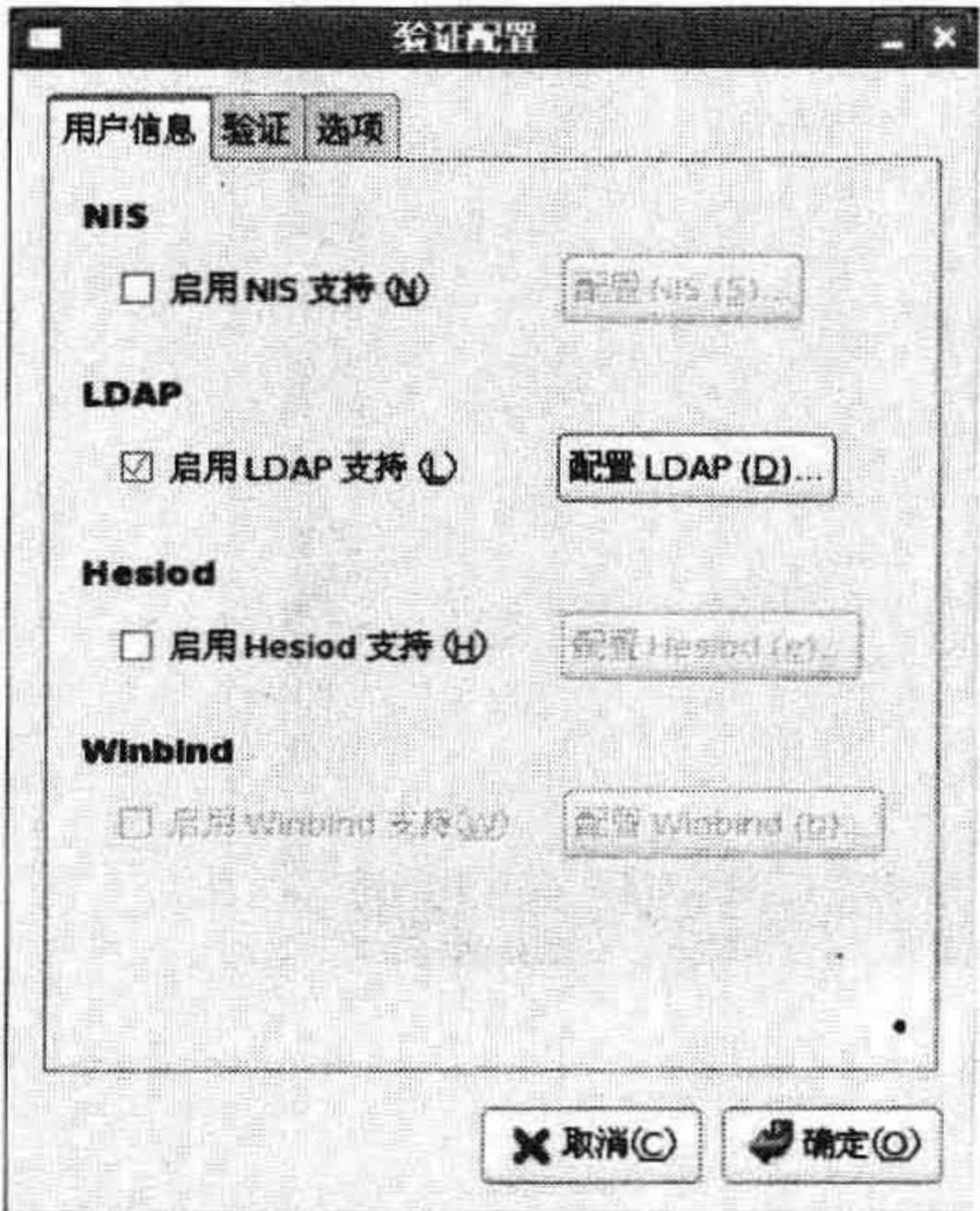


图 16-21 认证管理对话框—用户信息选项卡

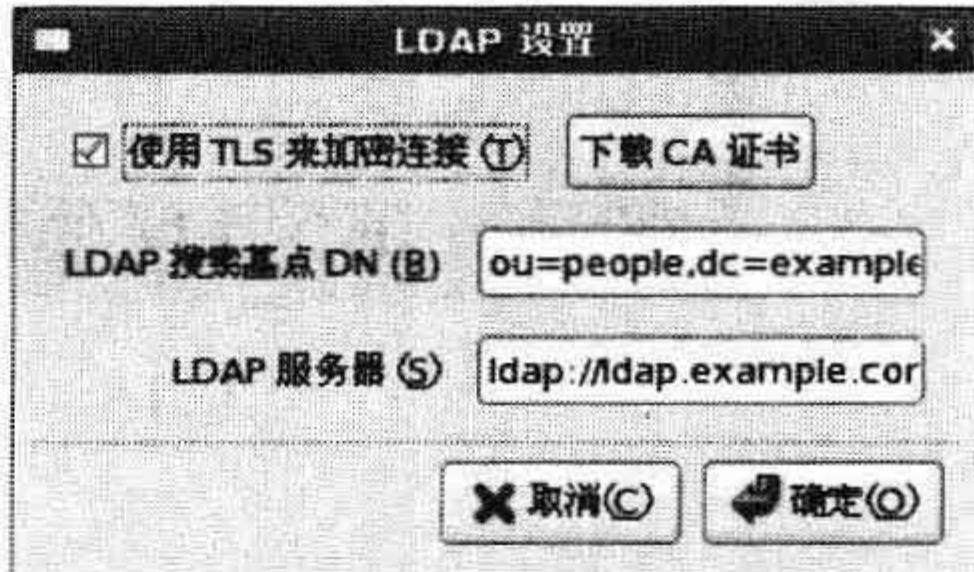


图 16-22 LDAP 配置

这样就定义了 LDAP 服务的详细信息，大家对此应该已经比较熟悉。由于已经指定需要使用 TLS 来加密链接，因此接下来要回答的问题就是 CA 证书的 URL 了（见图 16-23）。

可能在一个虚拟主机上已经有了一个可以快速下载并安装的 CA 证书，也或许还没有。如果正在使用的 CA 已经存在于本地的文件系统中，可以手动编辑 `/etc/ldap.conf` 文件告诉 PAM 在哪儿可以找到它。


```
tls_cacertfile /etc/pki/tls/certs/ca-bundle.cert
```

此文件与前面根据不同的发行版在/etc/ldap/ldap.conf 或/etc/openldap/ldap.conf 处所定义的文件是一样的。

在“验证”选项卡（见图 16-24）设定 LDAP 执行认证服务，这和指定用户信息时的过程是一样的，详细信息已填写。

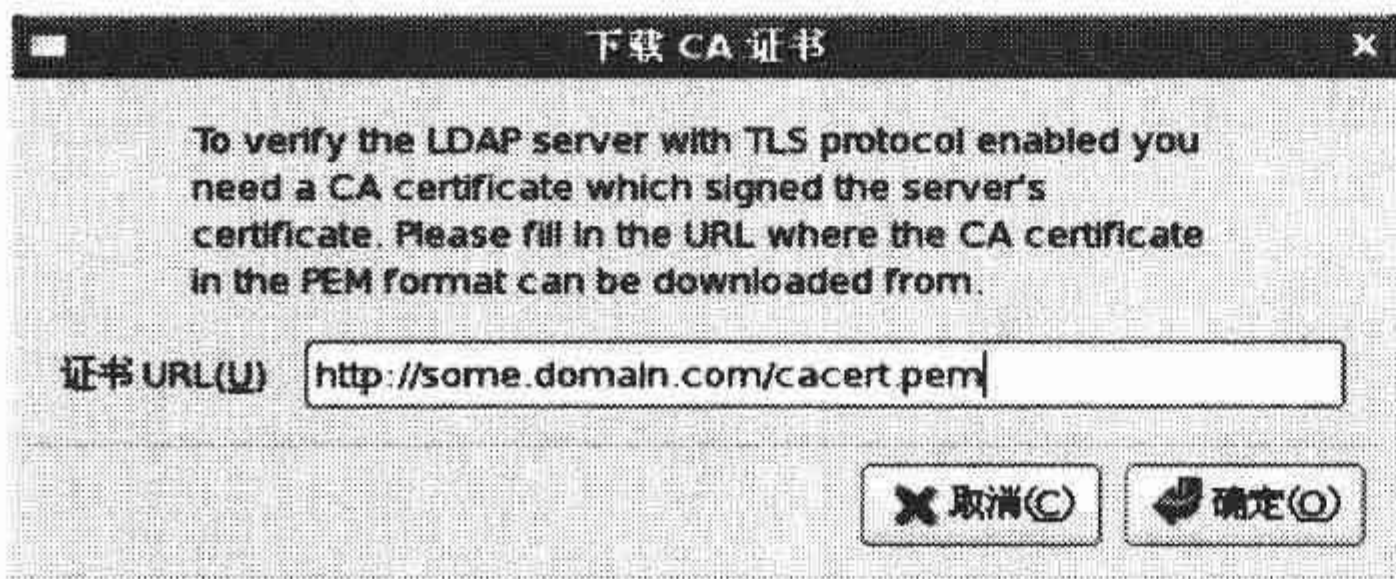


图 16-23 下载 CA 证书的 URL



图 16-24 验证配置对话框——“验证”选项卡

在“选项”选项卡中，如图 16-25 所示，设定 LDAP 服务的选项。和 Ubuntu 主机一样，选择“MD5”作为密码加密机制，指定本地用户可以只进行本地认证（允许本地用户不使用 LDAP 进行认证），同时指定用户首次登录时创建用户的 home 目录。

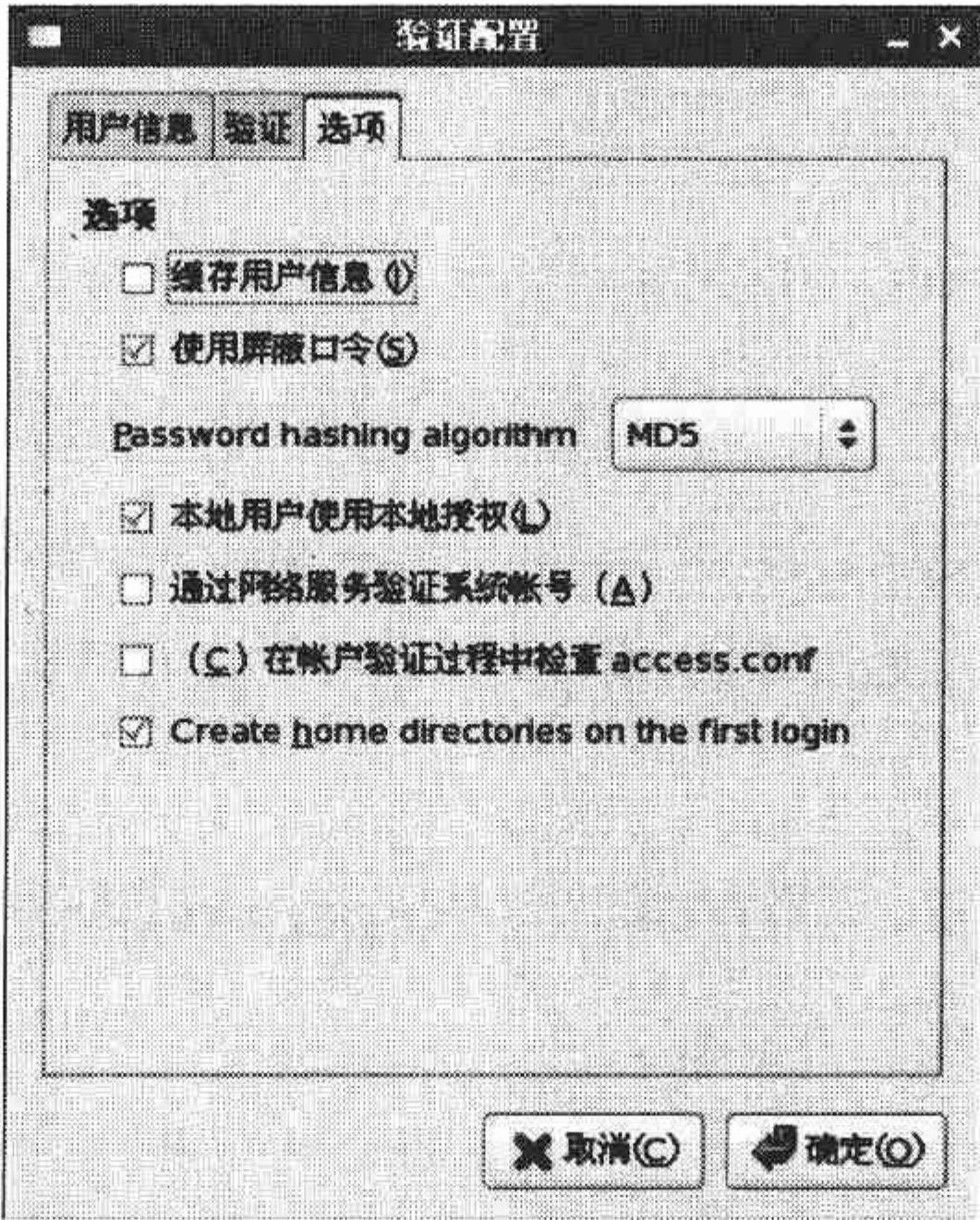


图 16-25 验证配置对话框——“选项”选项卡

16.7.2 PAM 运行机制

第 4 章提到过，Linux 可以使用插入式验证模块（PAM）通过 LDAP 服务运行认证服务。PAM 连接到 LDAP 服务器对主机提供了验证、授权以及更改密码的功能。PAM 使用位于/etc/pam.d

路径下的文件进行配置。就像第 4 章描述的一样，在 Red Hat 主机上，PAM 的主要文件是/etc/pam.d/system-auth 文件。列表 16-4 显示了在主机上建立 LDAP 认证的示例配置。

列表 16-4 Red Hat 主机上文件 system-auth 中的 PAM 设置

auth	required	pam_env.so
auth	sufficient	pam_unix.so nullok try_first_pass
auth	requisite	pam_succeed_if.so uid >= 500 quiet
auth	sufficient	pam_ldap.so use_first_pass
auth	required	pam_deny.so
account	required	pam_unix.so broken_shadow
account	sufficient	pam_localuser.so
account	sufficient	pam_succeed_if.so uid < 500 quiet
account	[default=bad success=ok user_unknown=ignore]	pam_ldap.so
account	required	pam_permit.so
password	requisite	pam_cracklib.so try_first_pass retry=3
password	sufficient	pam_unix.so md5 shadow nullok try_first_pass use_authtok
password	sufficient	pam_ldap.so use_authtok
password	required	pam_deny.so
session	optional	pam_keyinit.so revoke
session	required	pam_limits.so
session	optional	pam_mkhomedir.so
session	[success=1 ➡ default=ignore]	pam_succeed_if.so service in crond quiet ➡
use_uid		
session	required	pam_unix.so
session	optional	pam_ldap.so

该文件是自动生成的，除非有必要，一般不应对其进行修改。从列表 16-4 中可以看到，文件由 4 个独立的管理用户组构成：auth，account，password 和 session。

拿出其中的一行看一下，即 auth 管理组的一个例子。

auth sufficient pam_ldap.so use_first_pass

此用户组通常使用某种密码挑战反应机制对用户进行验证。sufficient 控制值表示如果该模块成功启用，则认为用户已通过验证。pam_ldap.so 是 PAM 要使用的共享对象。最后，use_first_pass 是可选语法项，表示此模块将使用在堆栈中较高级别模块里第一次提供过的密码，而不是再次询问密码。

在 Ubuntu 主机上，相应的文件分别是位于/etc/pam.d 路径下的 common-auth、common-password、common-session 和 common-account。

■注：在系统管理员指南中可以找到更多关于 PAM 的内容，见“http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/Linux-PAM_SAG.html”。

另一个使用 LDAP 服务进行 PAM 认证的核心文件是/etc/nsswitch.conf，此文件中要将 passwd、group 和 shadow 关键字设定为如下值。

passwd: files ldap
group: files ldap
shadow: files ldap

这些设置告诉 PAM 要使用哪些数据库以及使用的顺序，因此当搜索通常可以在 `/etc/passwd` 中找得到的信息时，将会先搜索主机上的文件，然后再搜索 LDAP，`group` 和 `shadow` 也一样。应该使用发行版提供的认证管理工具配置 PAM 和 `nsswitch.conf` 文件。

另一个需要注意的比较重要的文件是 `/etc/ldap.conf` 文件。需要设置一下 `/etc/ldap.conf` 文件的细节信息，里面包含主机中指向 PAM LDAP 模块的信息。Linux 主机可以使用很多不同类型的认证服务—LDAP、NIS 亦或活动目录。`/etc/ldap.conf` 文件中定义了诸如属性映射等在内的 LDAP 服务设置。

■注：当认证服务要求一个通常没有在 OpenLDAP 中提供的属性时，就要进行属性映射，比如 NIS 服务器请求某个属性时。作为一个例子，参考下列网址处是如何整合 Linux 和活动目录服务的，见“<http://www.linux.com/articles/40983>”。

通常在 `/etc/ldap.conf` 文件中包含了许多可以用来对主机进行认证的设置。列表 16-5 显示了用来实现为 LDAP 服务进行认证的基本配置。

列表 16-5 `/etc/ldap.conf`

```
base dc=example,dc=com
uri ldap://ldap.example.com/
scope sub
timelimit 120
bind_timelimit 120
idle_timelimit 3600
pam_filter exampleActive=TRUE
tls_cacertfile /etc/pki/tls/certs/ca-bundle.crt
ssl_start_tls
tls_cacertdir /etc/pki/tls/certs
pam_password md5
```

可以看到，这里也需要设置 LDAP DIT 的基点，即“`ou = People, dc = example, dc = com`”。`uri ldap` 指向 LDAP 主机。“`scope,bind timelimit`”以及“`idle timelimit`”都是默认设置，也可以对其进行修改以适应运行环境。“`pam filter`”可以是像“`(&(uid = <username>) (example Active = TRUE))`”一样的逻辑与或逻辑或搜索过滤器。这个搜索过滤器可以通过搜索索引的 `exampleActive` 属性来加快搜索速度，同时还能针对无效用户账号进行的未授权访问提供简单的保护。接下来，对 SSL 进行详细设置。“`tls cacertfile`”和“`tls cacertdir`”指向 Red Hat 主机上的目录，在 Ubuntu 主机上，应该指向 `/etc/ssl/certs` 目录。

现在可以测试一下是否能在经过 LDAP 认证的主机上访问用户账号，可以使用 `id` 工具或 `getent passwd` 工具进行此项测试。

`id` 工具输出用户的身份，同时还会搜索现有的所有用户数据库或文件来获取细节信息。在主机上选择一个不具有本地账户的用户执行如下操作。

```
$ id bsingh
uid=1003(bsingh) gid=1000(staff) groups=1000(staff)
```

另外一个方法是使用 `getent passwd`，它将会从管理数据库中获取如下所示的条目信息。

```
$ getent passwd
<snip>
```



```
openldap:x:117:128:OpenLDAP Server Account,,,:/var/lib/ldap:/bin/false
bacula:x:118:129:Bacula:/var/lib/bacula:/bin/false
nobody :x!:65534:65534:nobody:/dev/null:
jsmith :*:1000:1000:Jane Smith:/home/jsmith:
bsingh:*:1003:1000:Bevan Singh:/home/bsingh:
ataylor:*:1002:1000:Angela Taylor:/home/ataylor:
jbob:*:1004:1000:Jim Bob:/home/jbob:
```

这是搜索返回结果的一个简表。账户 `openldap` 和 `bacula` 是本地用户账号，从 `nobody` 到 `jbob` 是 LDAP 服务器提供的账号。

接下来要做的事情是使用其中的一个账号登录主机。进行此项测试的一个简单方法是使用 SSH。

```
# ssh jbob@localhost
jbob@localhost's password:
Linux ldap.example.com 2.6.24-23-server #1 SMP Mon Jan 26 00:55:21 UTC 2009 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
Last login: Fri May 1 01:05:46 2009 from localhost.localdomain
$
```

这个例子中，尽管 `jbob` 用户在主机上并没有账号，他还是使用 SSH 登录到了 `localhost` 主机。现在就可以使用 `jbob` 或在 LDAP 服务器中的其他任何有效用户账号登录 LDAP 服务器所认识的所有主机了。

16.7.3 LDAP 和 Apache 认证

现在来看看如何使 Web 服务器也能使用 LDAP 服务器对客户端进行认证。当客户端尝试访问“`https://ldap.example.com`”站点时，将会被要求先填写 LDAP 用户名和密码，然后才能获准访问。为了达到这个效果，需要对 Web 服务器进行两项设置：一是在 Web 主机上启用 SSL 以确保所有与 Web 服务器的链接的安全性，另一个是在 `ldap.example.com` 虚拟主机中添加 LDAP 详细信息。

■注：第 11 章讨论了 Apache 虚拟主机。

首先假设这是正在运行的 `ldap.example.com` 主机，而且其上没有其他的 Apache 服务运行。首先，在 Ubuntu 主机上，将以下内容添加到 `/etc/apache2/httpd.conf` 文件中。

```
LDAPTrustedGlobalCert CA_BASE64 /etc/ssl/certs/ldap.pem
LDAPTrustedMode TLS
LDAPVerifyServerCert Off

NamedVirtualHost 192.168.0.1:80
NamedVirtualHost 192.168.0.1:443
```


■注：同样，可以将这些指令添加到 Red Hat 主机的/etc/httpd/conf.d/httpd.conf 文件（此文件需手动创建）中，或者将其放在虚拟主机文件的顶部。

接下来，检查一下将要改变的 ldap.example.com 虚拟主机文件。

```
$ sudo vi /etc/apache2/sites-available/ldap.example.com
<VirtualHost 192.168.0.1:443>
    ServerName ldap.example.com
    DocumentRoot /var/www/sites/lam

    SSLEngine on
    SSLCertificateFile /etc/ssl/certs/ldap.pem
    SSLCertificateKeyFile /etc/ssl/certs/ldap.pem
    LogFormat "%v %l %u %t \"%r\" %>s %b" comonvhost
    CustomLog /var/log/apache2/vh_access.log comonvhost
    ErrorLog /var/log/apache2/vh_error.log
    LogLevel debug

Alias /lam /usr/share/ldap-account-manager

<Directory /usr/share/ldap-account-manager>
    Options +FollowSymLinks
    AllowOverride All
    Order allow,deny
    Allow from all
    AuthType Basic
    AuthName "LDAP example.com"
    AuthBasicProvider ldap
    AuthzLDAPAuthoritative on
    AuthLDAPURL ldap://ldap.example.com/ou=people,dc=example,dc=com?uid?sub
    AuthLDAPBindDN cn=Webadmin,ou=meta,dc=example,dc=com
    AuthLDAPBindPassword Zf3If7Ay
    Require valid-user
    Require ldap-group cn=admin,ou=Groups,dc=example,dc=com
    DirectoryIndex index.html
</Directory>
...
<snip>
...
</VirtualHost>
```

■注：在 Red Hat 主机上，此文件位于/etc/httpd/conf.d/vhost.conf，其位置取决于对 Red Hat 虚拟主机的管理方式。

添加如下代码来启用 Web 服务器的 LDAP 认证。

```
LDAPTrustedGlobalCert CA_BASE64 /etc/ssl/certs/ldap.pem
LDAPTrustedMode TLS
LDAPVerifyServerCert Off
```

在这里，设定了证书文件和认证模式，同时设定拒绝验证服务器证书。“LDAPTrustedMode TLS” 设定所有的 LDAP 链接使用 TLS 安全方式传输。上述指令必须位于<VirtualHost>标签外部、<VirtualHost>标签内部，在其中需要添加如下代码。


```
AuthType Basic
AuthName "LDAP example.com"
AuthBasicProvider ldap
AuthzLDAPAuthoritative on
AuthLDAPURL ldap://ldap.example.com/ou=people,dc=example,dc=com?uid?sub
AuthLDAPBindDN cn=Webadmin,ou=meta,dc=example,dc=com
AuthLDAPBindPassword Zf3If7Ay
Require valid-user
Require ldap-group cn=admin,ou=Groups,dc=example,dc=com
```

上面把 `AuthType` 设为 “Basic”，把 `AuthName` 设为 “LDAP example.com”。`AuthType` 定义的是认证方式，有 Basic 和 Digest 两种选择。LDAP 认证要使用 Basic。`AuthName` 就是在弹出的认证窗口中的名称。

“`AuthBasicProvider ldap`” 定义了要用来提供认证机制的服务器，在这里就是 LDAP 服务器。指定 “`AuthzLDAPAuthoritative on`” 表明希望使用 LDAP 服务器提供的认证服务允许或拒绝访问请求。接下来是将要使用的认证服务器的 LDAP URL，设为 “`AuthLDAPURL ldap://ldap.example.com/ou=people,dc=example,dc=com?uid?sub`”。这里指定搜索的基点为 “`ou=people,dc=example,dc=com`”，搜索所感兴趣的属性为 `uid`，搜索的范围是 `sub`。现在，可以在这里看到元账号 “`cn=Webadmin,ou=meta,dc=example,dc=com`” 用到何处，而且这个账号将会使用已经提供的密码绑定到 LDAP 服务器。

最后指定需要一个 `valid-user`，而且认证中的用户必须属于 LDAP 用户组 “`cn=admin,ou=Groups,dc=example,dc=com`”。

■注：关于 LDAP 和 Apache 认证的更多信息，可以参考 “http://httpd.apache.org/docs/2.2/mod/mod_ldap.html”。

开始之前，还要确保所有必须的模块都已经安装到了 Web 主机中，在 Ubuntu 中可以执行以下命令。

```
$ sudo a2enmod authnz_ldap
$ sudo a2enmod ssl
```

在 Red Hat 中，需要确保文件 `/etc/http/conf/httpd.conf` 中存在下列代码行，同时确保已经安装了 `mod_ssl` 软件包。

```
LoadModule auth_basic_module modules/mod_auth_basic.so
LoadModule ldap_module modules/mod_ldap.so
LoadModule ssl_module modules/mod_ssl.so
LoadModule authnz_ldap_module modules/mod_authnz_ldap.so
```

可以使用下面的命令在 Ubuntu 中启动 Web 服务。

```
$ sudo /usr/sbin/invoke-rc.d apache2 restart
```

在 Red Hat 中，命令如下所示。

```
$ sudo /sbin/service httpd restart
```

现在使用 Web 浏览器在以下地址链接到 LAM Web GUI: <https://ldap.example.com/lam>。

从图 16-26 中，可以看到 Apache 提供的身份验证挑战。其中填入了名为 ataylor 的 uid，此 uid 隶属于用户组 “cn = admins,ou = Groups,dc = example,dc = com”，而且是 Apache 配置所必须的成员。



图 16-26 Apache 要求输入用户名和密码

在 Apache 日志文件中将会出现如下内容。

```
[Sun Apr 05 16:31:23 2009] [info] Initial (No.1) HTTPS request received for ➡
child 3 (server ldap.example.com:443)
[Sun Apr 05 16:31:23 2009] [debug] mod_authnz_ldap.c(377): [client 192.168.0.2] ➡
[1807] auth_ldap authenticate: using URL ➡
ldap://ldap.example.com/ou=people,dc=example,dc=com?uid=sub, ➡
referer: https://ldap.example.com/lam/templates/login.php
[Sun Apr 05 16:31:23 2009] [debug] mod_authnz_ldap.c(474): [client 192.168.0.2] ➡
[1807] auth_ldap authenticate: accepting ataylor, referer: ➡
https://ldap.example.com/lam/templates/login.php
[Sun Apr 05 16:31:23 2009] [debug] mod_authnz_ldap.c(715): [client 192.168.0.2] ➡
[1807] auth_ldap authorise: require group: testing for group membership in ➡
"cn=admins,ou=Groups,dc=example,dc=com", referer: ➡
https://ldap.example.com/lam/templates/login.php
[Sun Apr 05 16:31:23 2009] [debug] mod_authnz_ldap.c(721): [client 192.168.0.2] ➡
[1807] auth_ldap authorise: require group: testing for member: uid=ataylor, ➡
ou=People,dc=example,dc=com (cn=admins,ou=Groups,dc=example,dc=com), referer: ➡
https://ldap.example.com/lam/templates/login.php
[Sun Apr 05 16:31:23 2009] [debug] mod_authnz_ldap.c(730): [client 192.168.0.2] ➡
[1807] auth_ldap authorise: require group: authorisation successful (attribute ➡
member) [Comparison true (cached)][Compare True], referer: ➡
https://ldap.example.com/lam/templates/login.php
```

这就表明，LDAP 服务器正在使用用户名 ataylor 验证访问请求，同时测试出此用户隶属于用户组 “cn = admins,ou = Groups,dc = example,dc = com”。在虚拟主机中给 loglevel 指令指定 debug 值就可以提供此等级的日志细节内容。

16.7.4 LDAP 与知识树 DMS 整合

在本章开始的时候提到过，现在有很多应用程序都是用 LDAP 提供的认证服务。其中之一就是在第 12 章中介绍过的 DMS 应用程序。下面快速了解一下如何在这样一个应用程序中详细配置 LDAP。

首先必须确保 DMS 主机与 LDAP 服务器可以有效交流。可以通过如下代码使用命令行工具进行测试。

```
ldapsearch -xvW -D cn=Webadmin,ou=meta,dc=example,dc=com -Z ➡
-h ldap.example.com -b ou=People,dc=example,dc=com uid
```

若在 DMS 主机上尝试使用此命令后有返回信息，那就说明主机可以和 LDAP 服务器交

流。否则，可能需要解决几个问题。第一个问题，要确定/etc/ldap/ldap.conf (Ubuntu 中) 或/etc/openldap/ldap.conf (Red Hat 中) 文件进行了正确的配置，这两个文件是 LDAP 客户端的本地配置文件，以下是示例文件中的详细配置代码。

```
URI ldap://ldap.example.com
BASE dc=example,dc=com
TLS_REQCERT allow
```

需要确定已经把 ldap.conf 文件复制到了 Web-root/knowledgetree-installation/common/etc/openldap 路径下，其中 Web-root 是安装知识树 DMS 的位置。示例中，它被安装到了/data/Web/DMS/common/etc/openldap 路径下。

假设已经登录并转到了 DMS 管理选项卡页面。当然，只有管理员才能打开此页面。现在试着添加一个新的认证源（见图 16-27）。

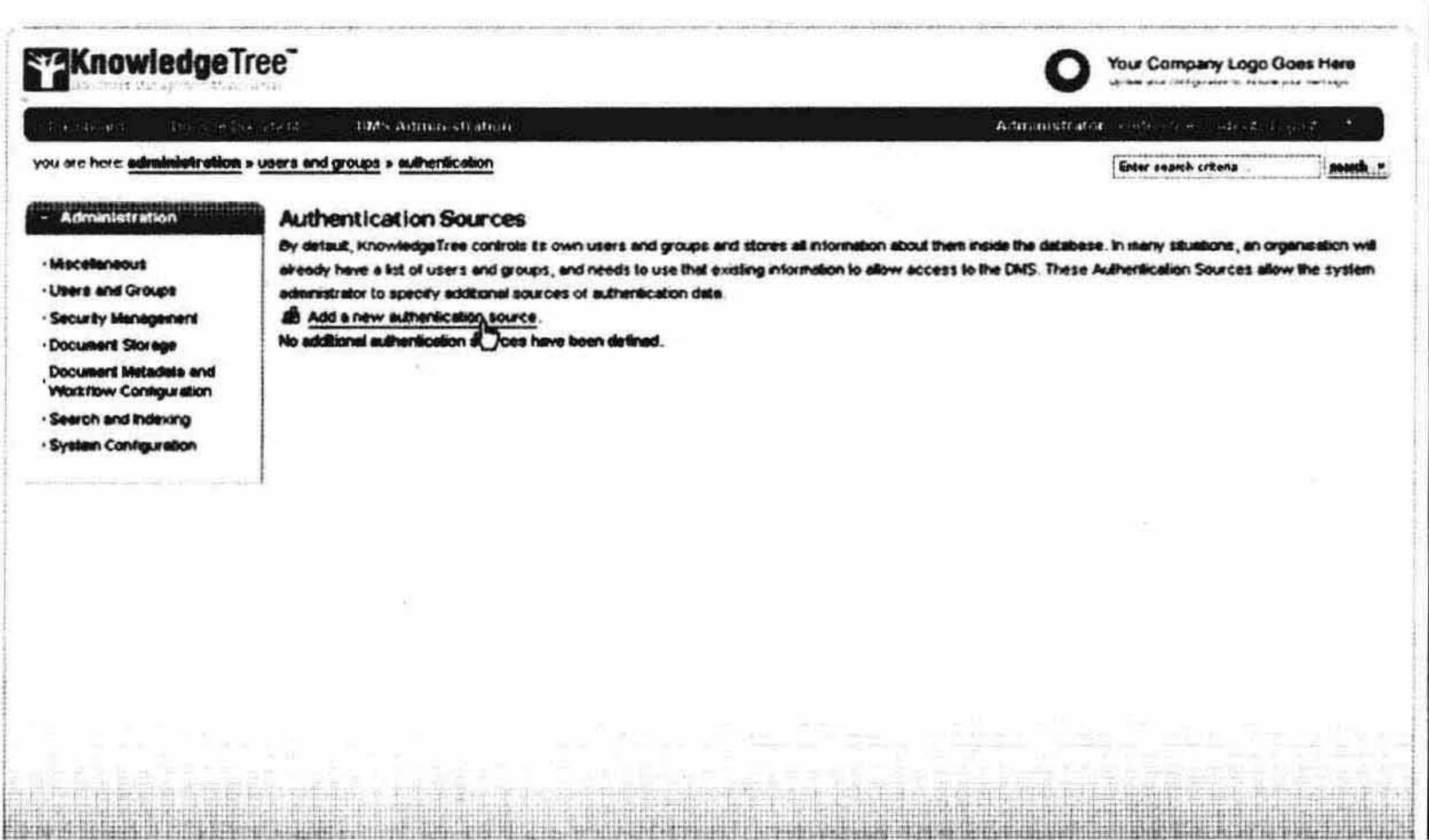


图 16-27 添加一个新认证源

单击链接后转到图 16-28 所示的页面，在此页面中确定要使用的认证源类型。

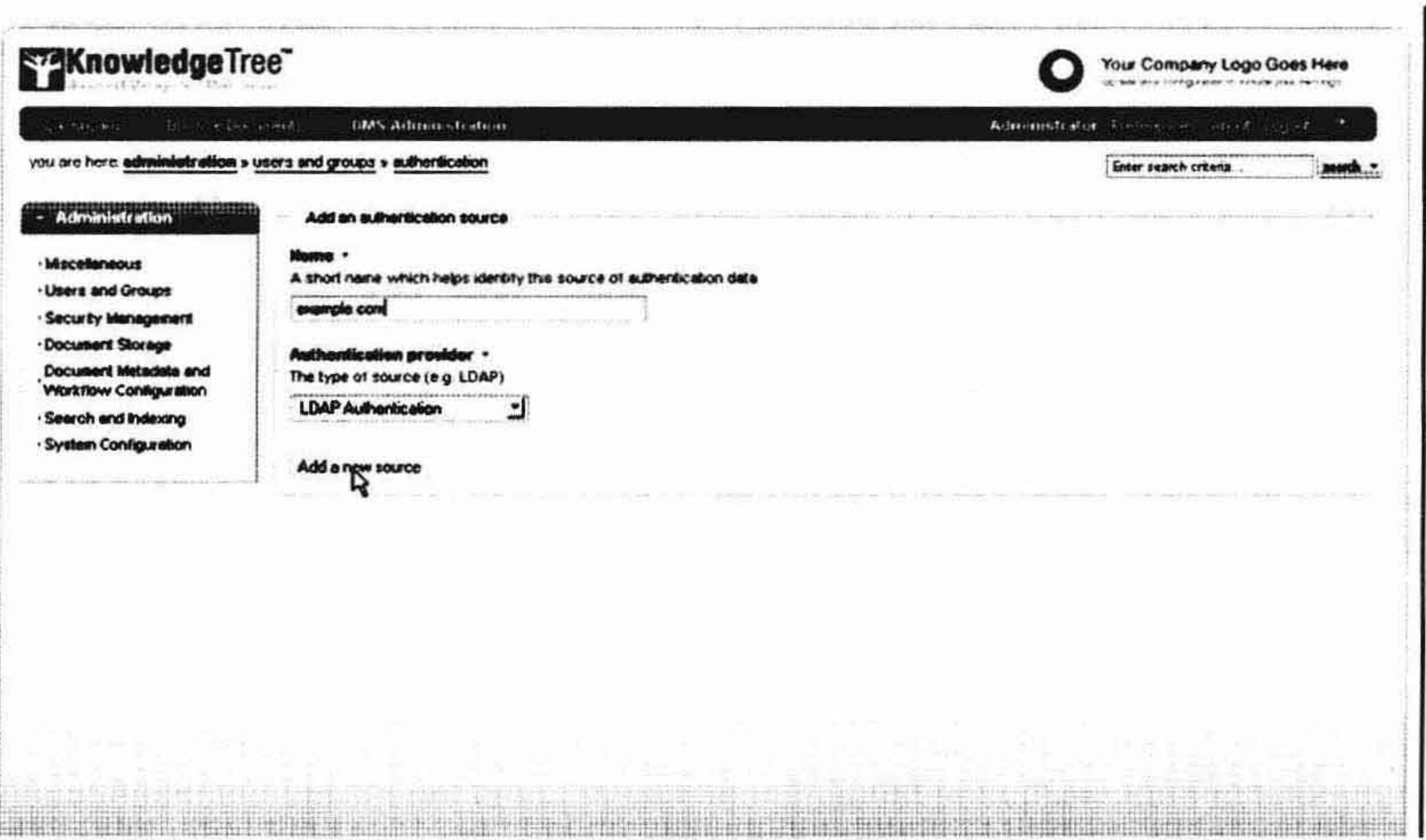


图 16-28 选择 LDAP 认证

名称设置 (Name setting) 用来识别身份验证服务，还可以在这里指定身份验证服务的提供程序。单击 “Add a new source” 转到下一页，如图 16-29 所示。

Security Management

Document Storage

Document Metadata and Workflow Configuration

Search and Indexing

System Configuration

Server name *

The host name or IP address of the LDAP server

ldap.example.com

Server Port *

The port of the LDAP server (default: 389)

389

Use Transaction Layer Security (TLS) *

Whether to use Transaction Layer Security (TLS), which encrypts traffic to and from the LDAP server

☒

Base DN *

The location in the LDAP directory to start searching from (CN=Users,DC=mycorp,DC=com)

ou=People,dc=example,dc=com

Search User *

The user account in the LDAP directory to perform searches in the LDAP directory as (such as CN=searchUser,CN=Users,DC=mycorp,DC=com or searchUser@mycorp.com)

cn=webadmin,ou=meta,dc=example,dc=com

Search Password *

The password for the user account in the LDAP directory that performs searches

Search Attributes *

The LDAP attributes to use to search for users when given their name (one per line, examples: cn, mail)

uid

图 16-29 LDAP 详细设置

在此页面上定义服务器名 (ldap.example.com)、服务端口 (389) 以及 TSL 加密。Base DN 设置为 “ou = People,dc = example,dc = com”。需要有一个认证 Web 服务的用户 “cn = webadmin, ou = meta, dc = example,dc = com”，其密码也定义在这里。还定义了搜索属性 “uid”，此属性已索引，使得可以以最快的速度找到用户。因为是用户的通用对象类，所以要对 person 对象类进行过滤（此项在图 16-29 中没有）。把上述详细设置保存后，就可以看到如图 16-30 所示的摘要页面了。

you are here: administration > users and groups > authentication > example.com (viewing)

Enter search criteria

search

Administration

Miscellaneous

Users and Groups

Security Management

Document Storage

Document Metadata and Workflow Configuration

Search and Indexing

System Configuration

Configuration updated

Standard configuration

Name

example.com

Provider

LDAP authentication provider

Edit standard configuration

Provider configuration

LDAP Server

ldap.example.com

The LDAP server port

389

Base DN

ou=People,dc=example,dc=com

LDAP Search User

cn=webadmin,ou=meta,dc=example,dc=com

LDAP Search Password

*** Hidden ***

Search Attributes

uid

Object Classes

person

posixAccount

Use Transaction Layer Security (TLS)

True

Edit provider configuration

图 16-30 认证服务设置摘要页

现在可以到用户管理页面在“Add a user from an authentication source”框中单击“Add from source”按钮，如图 16-31 所示。

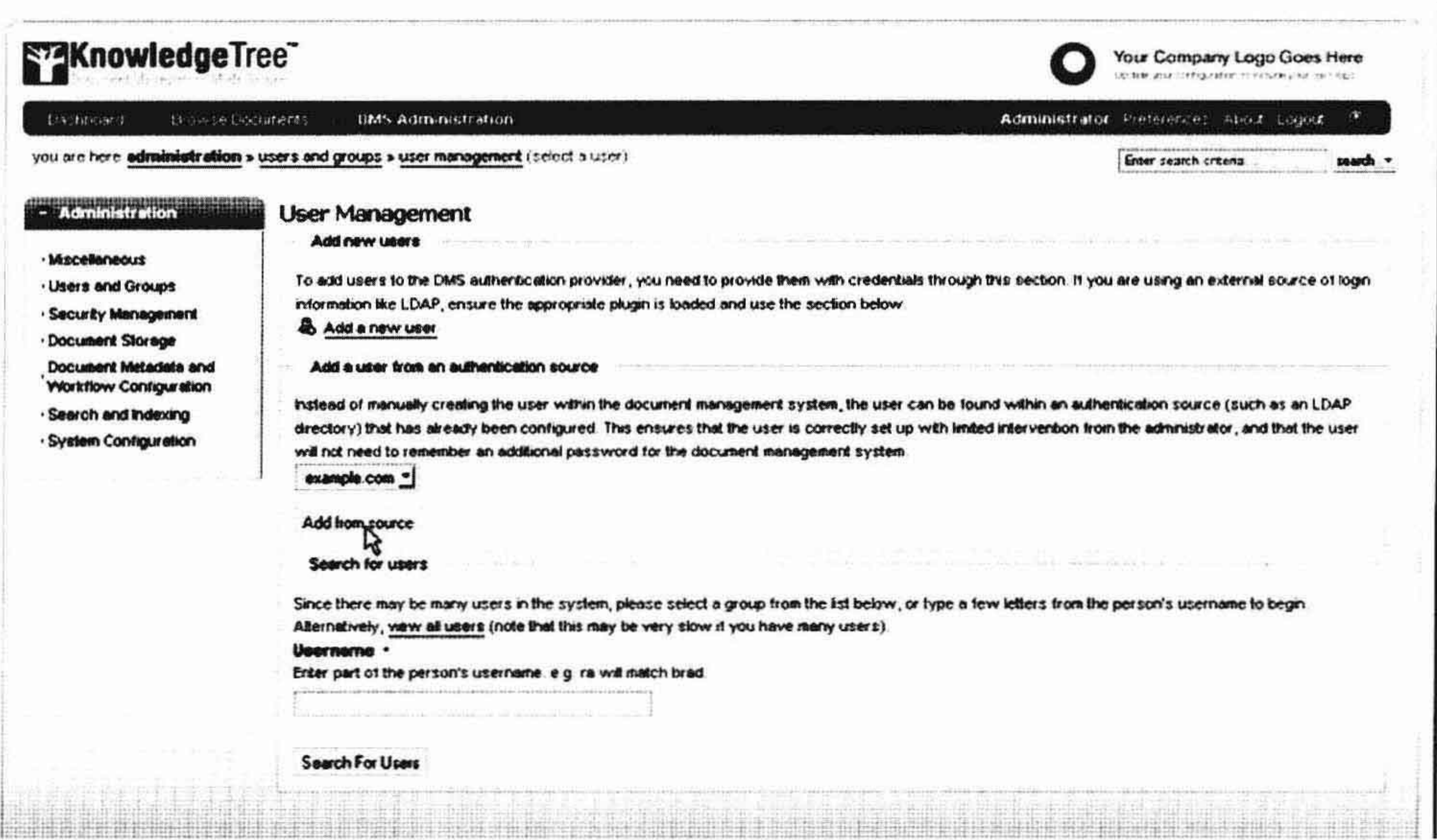


图 16-31 从源添加用户

下一步，要在 LDAP 目录中搜索一个用户或批量导入多个用户。如图 16-32 所示，对用户 jbob 进行搜索。

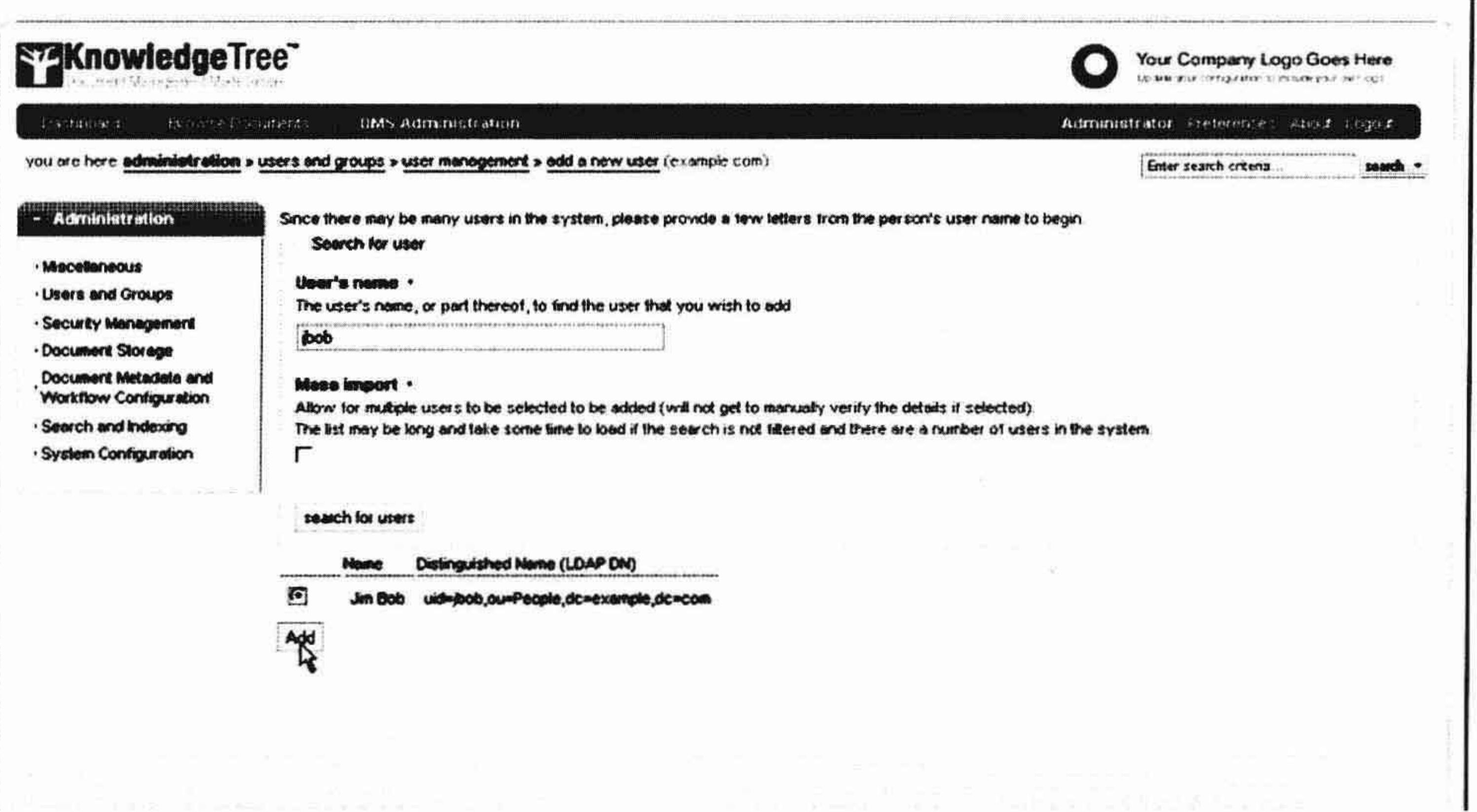


图 16-32 添加用户 jbob

单击“search for users”按钮后，系统返回了用户的相关信息。下面要做的就是添加 Jim Bob 用户，然后他就可以和其他普通用户一样在系统中开始工作了。

■提示: 完成所有的安装和调试工作后, 要记得把 `slapd.conf` 文件中的 `LogLevel` 设置修改为 0, 然后重启 `slapd` 服务, 使其工作在实际使用模式下。

16.8 小 结

本章中, 讨论了什么是目录服务器, 以及条目在目录信息树中是如何组织的。演示了如何配置和安装一个 OpenLDAP 目录服务器并在其中添加用户账号和管理员账号。讨论了模式、索引以及访问控制列表。演示了如何使用 OpenLDAP 提供的各种各样的工具来对 LDAP 服务器进行管理和查询。现在也可以配置一个基于 Web 的 GUI 来管理 LDAP 目录, 也可以把 LDAP 整合到网络以及现有的应用程序中去。

经过本章的学习后, 读者应该可以进行如下的操作了。

- 在 Ubuntu 和 Red Hat 中安装和配置 OpenLDAP。
- 理解与配置访问控制列表。
- 查询与管理 LDAP 目录。
- 安装与配置 LAM Web GUI。
- 在 Linux 上为 LDAP 设置单点登录。
- 配置 Apache Web 服务器以使用 LDAP 认证服务对客户端访问进行认证。
- 整合 OpenLDAP 与知识树 DMS。
- 综上所述, 目录服务可以在网络中承担管理中心的角色, 还有很多关于目录服务的内容在本章中都没有涉及。建议读者买一本专门讨论此内容的参考书, 或者阅读一下链接 “<http://www.openldap.org>” 处的在线文件, 也可以使用邮件列表来帮助深入了解这方面的知识。

下一章讨论性能监控和优化方面的内容。

第 17 章 性能监控与优化



Peter Lieverdink 编写

现在主机已经在运行着很多服务了，而且更重要的是它会一直持续工作。当公司的业务量增长时，服务器上所加载的工作也会越来越多。本章将会讲述如何对硬盘空间和处理器及 RAM 的使用率等资源进行监控，这将有助于识别性能瓶颈，并更好地利用各种资源。

由于通过进行少量的调整就可以缓解购买额外硬件的需求，因此性能检测可以在一定程度上为使用者节省金钱和时间。

17.1 基本的健康状况检查

本节介绍一些可以帮助确定主机状况的基本工具。

17.1.1 CPU 利用率

当应用程序运行变慢时，首先要检查系统是否真正地正在忙于什么事情。进行检查的最快的方法是使用 `uptime` 命令，如列表 17-1 所示。

列表 17-1 `uptime` 命令

```
$ uptime
12:51:04 up 116 days, 7:09, 3 users, load average: 0.01, 0.07, 0.02
```

这个命令输出了系统运行状态的一个很短的概述，包括当前时间、系统运行时间（或开机时间）、登录的用户以及显示系统负荷的 3 个数字。

此例中，系统最后一次启动是在 116 天 7 小时 9 分钟之前。如果与远程主机断开了链接，凭此信息可以很方便地判断出系统是否已经重新启动以及能否重新登录。

已登录用户的数目是指目前使用终端控制台、使用 X Window 以及使用 SSH 登录的所有用户的总数。如果一个用户登录了两次，将被视为是两个用户，其中不算登入诸如 Samba 或 Apache 等某种服务的用户。

最后，系统负载用 3 个数字来显示。这些数字以索引的形式显示 CPU 正在运行的工作与过去 1 分钟、5 分钟和 15 分钟时间里所排定工作的平均比率。

1.00 的负荷意味着所排定的工作与 CPU 能够处理的最大工作量相同。然而，如果一个主机有多个 CPU，则应将每一个 CPU 的负荷取和，也就是说，对于一个具有 4 个 CPU 的主机来讲，4.00 的负荷才和单 CPU 系统的 1.00 负荷是一样的。

在示例中，过去 1 分钟内的平均负荷大约是 1%，而过去 5 分钟内的平均负荷时 7%，这就说明，有一些其他的任务运行过，并且在几分钟前停止了。

尽管这些数字并不能被看作是解决问题的万能钥匙，但它们也的确提供了一个判断系统健康与否的快速检查方法。如果在具有 8 个 CPU 的主机上的负荷是 6，那可能有些任务刚刚运行结束。但如果其负荷是 60，那主机肯定有问题了。

问题可能是主机正在运行几个垄断 CPU 的进程，从而不能给其他进程足够的处理周期。稍后将会看到如何使用 top 工具找出这些垄断进程。当然也可能是主机正在尝试运行超出其处理能力的更多进程，这也会导致没有足够的 CPU 周期用来处理其他进程。写的不好的应用程序或守护进程不断再生复制自身时就会导致这种状况。更多信息参见下面“fork 炸弹”部分的介绍。稍后将会学习如何使用 ulimit 命令防止此类问题的发生。

最后一种情况，也可能进程正在等待主机运行一个不能被中断的任务，比如对硬盘进行数据写入或读出操作。如果进程继续进行之前，此任务必须完成，则称进程进入了非中断性休眠状态。这种状态下的进程将会使平均负荷持续增加，但是进程并没有真正地使用 CPU。稍后在“高级工具”部分将会看到如何检查硬盘的使用状况。

fork 炸弹

在第 5 章讨论过，当一个守护进程或服务在后台运行时，它产生一个子进程并初始化父进程，同时，原始进程退出。一个有问题的应用程序运行出错时，非常有可能不断地产生子进程，而从不退出，这就叫 fork 炸弹。

一个 fork 炸弹会以系统所允许的速度持续不断地复制产生其自身。1 个产生 2 个，2 个产生 4 个，4 个产生 8 个，如此重复不断，最终使其他进程得不到所需的 CPU 周期和 RAM 资源。

17.1.2 内存利用率

过度的内存使用是造成系统性能降低的另一个原因。如果一个任务使用内存过多，主机为了满足其需要就不得不把其他的任务转存到交换内存中。交换内存毕竟只是一块当做内存使用的硬盘空间，对交换内存的存取比 RAM 要慢得多。

■注：依据 BIOS 和系统内核的不同，一个 32 位的操作系统一般只能访问 2GB 到 4GB 的 RAM。建议在服务器主机上安装 64 位版本的 Linux，这样就没有上面的访问限制了。

可以使用 free 命令快速地查看一下主机上使用的 RAM 和交换内存的数量。如列表 17-2 所示，在命令后加上-m 选项以兆字节（MB）为单位显示空间大小，而不是千字节（KB）。

列表 17-2 free 命令

```
$ free -m
```

	total	used	free	shared	buffers	cached
Mem:	503	147	356	0	17	56
-/+ buffers/cache:		72	430			
Swap:	23	0	223			

此列表给出了系统运行的可用内存是否过低的即时概述。第一行显示 RAM 使用状况，total 指主机总可用内存。接下来把总内存分成了已用内存和可用内存两部分。

shared 列已经不再使用了，但仍然显示了出来，以防止依赖此输出格式的工具崩溃。

buffers 列显示内核用作磁盘写缓存的内存数量。此缓存允许应用程序快速写入数据，由内核负责在后台将数据写入磁盘。也可以从此缓存中读数据，以提供额外的速度提升。最后一列（cached）显示内核用作高速缓存的内存数量，高速缓存（就像 Squid）提供信息的高速存取。

缓存和高速缓存都可以根据需要调整大小。如果一个应用程序需要更多的 RAM，内核将腾出部分缓存或高速缓存空间，并对内存进行重新分配。

第二行也显示了已用和可用的内存数量，但内存总数中不包括缓存和高速缓存，因此，此信息对于目前应用程序实际使用的内存数量给出了更准确的描述。

最后一行显示主机使用了多少交换空间。随着时间的推移，一部分不再使用的服务可能会被内核转存到交换空间中，因此这个数目会逐渐地增加。这样做便于回收闲置的内存空间以用作缓存或高速缓存。

这意味着让主机使用交换空间并不一定是坏事。然而，如果所有的内存和所有的交换空间都被占用了，显然就会有问题了。

■注：在 Windows 中，确保主机有可用内存空间来运行应用程序或许是可取的，然而在 Linux 中就不是这么回事了。可用内存空间意味着是浪费的内存空间，因为它没有被利用起来，甚至都没用来做缓存或高速缓存。

在“高级工具”部分将会看到如何查看正在使用的独立任务内存有多少。

17.1.3 磁盘空间

计算机系统的另一个有限资源是磁盘空间和磁盘速度。一般来讲，磁盘空间已满不会使系统运行变慢，但却会使服务崩溃而引起用户的不满，因此，对磁盘的使用进行监控并在有需要时提供更多可用磁盘空间是值得做的。

下一章将讨论如何检查磁盘的速度问题。

■注：在第 8 章中已经讲了使用 df 和 du 工具检查可用和已用磁盘空间的内容。

17.1.4 日志

最后，如果应用程序或内核发生了某些始料未及的问题，在系统或内核的日志中找到相

关的条目将是非常有用的。崩溃的服务当然需要重启，但查看日志了解问题发生的原因或将有助于防止同样的事情再次发生。

如果日志守护进程自己出问题停止了，还可以使用 `dmesg` 命令查看内核日志缓存。

注：第 18 章将详细讨论日志的相关内容。

17.2 高级工具

基础工具能给出一个快速概述，但若出现问题，却不能提供帮助确定问题原因的信息。为此，这里将介绍一些能帮助确定瓶颈的工具。

17.2.1 CPU 和内存利用率

Linux 提供了 `top` 工具来列出正在运行任务的详细信息。它和在 windows 系统中所使用的任务管理器很相似，但运行在终端窗口中，如图 17-1 所示。它提供了一个主机中正在运行的进程和线程的可排序、可管理的列表。

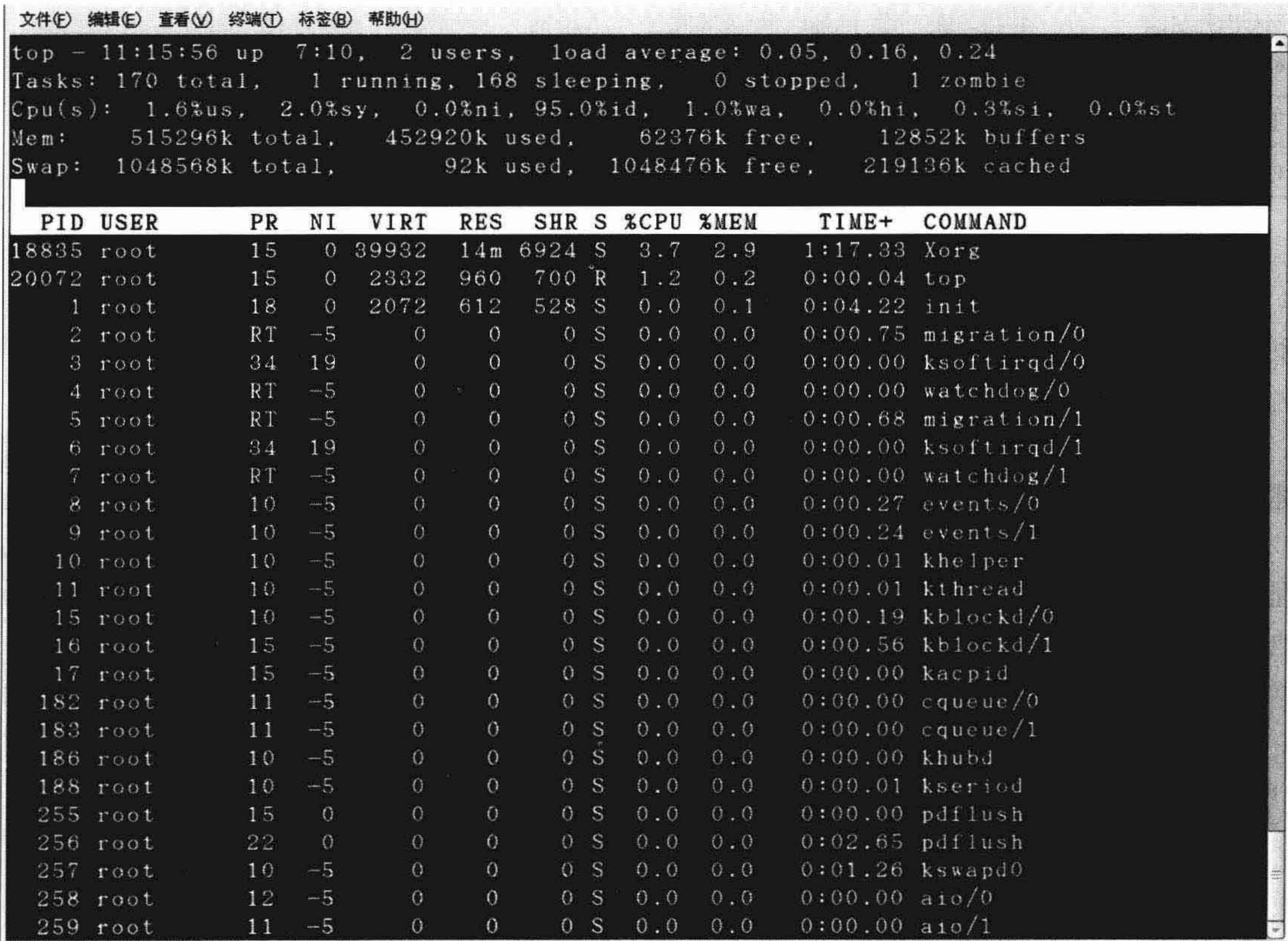


图 17-1 top 工具

输出的顶部是几个标题行，包括了命令 `uptime` 中的信息，以及从 `free` 命令和 `vmstat` 命令中来的汇总数据，这两个命令会在“交换空间利用率”和“读写磁盘”部分进行讨论。可以对这些标题的显示与否进行切换。使用“L”键切换平均负荷行的显示与否，使用“T”键切换任务汇总信息的显示与否，使用“M”键切换内存信息的显示与否。

剩下的显示信息包括了几列关于正在运行的进程和线程的信息。这里显示的是默认的列，还可以显示或隐藏其他的列。如表 17-1 所示是一些信息的标题及其意义。

表 17-1 top 命令的列标题	
标题	意义
PID	任务进程 ID。这一唯一的标识符允许对任务进行相关操作
USER	任务所有者的名称，运行任务的账号
PR	任务优先级
NI	任务友好度，表示任务是如何将 CPU 周期出让给其他任务的。低的或负的友好度意味着高优先级
VIRT	任务使用的内存总数，包括共享内存和交换空间
RES	任务使用的所有物理内存总数，不包括交换空间
SHR	任务使用的共享内存总数。此内存由库函数分配并同时被其他任务所使用
S	任务状态。表示任务正在运行（R）、休眠（D 或 S）、挂起（T）或僵住（Z）
%CPU	自最后一次屏幕刷新以来此任务所用的有效 CPU 周期百分比
%MEM	任务所用有效内存百分比
TIME+	自任务启动以来所占用的 CPU 时间
COMMAND	监视的任务名称

在 `man top` 手册页可以获得以上及其他所有可用字段的详细描述。

默认的任务显示方式是用任务状态排序并按降序排列，这就意味着如果一个任务挂起或僵住（例如运行有问题），那它就会排在列表的顶部。参考“僵住：不死的任务”栏来理解什么是僵住的进程。

为了使排序显示结果对检查 CPU 利用率更有用，需要按 `%CPU` 列进行排序。按下“F”键，调出所有列标题的列表，在这里可以指定输出信息按哪一列进行排序，如图 17-2 所示。

可以按下“k”键使 `%CPU` 列作为排序字段，然后按回车键回到任务列表。现在所有进程就按所使用的 CPU 周期总数进行排列了，因此，如果有一个进程肆意横行，就可以很容易地将它定位。

僵住：不死的任务

除了 `init` 进程，Linux 主机上的每一个任务都由其父进程所控制，当其运行结束时还要向父进程报告其状态。

然而有时事情会出问题，父进程也可能会崩溃，留下一个运行完成的子进程在等待报告其状态。这种情况发生时，子进程就进入了僵住状态——一个没有控制者的死（运行完成）任务。通常经过很短的时间这类进程会自己解困并退出。

更多关于僵住状态的信息，请参考“http://en.wikipedia.org/wiki/zombie_process”。

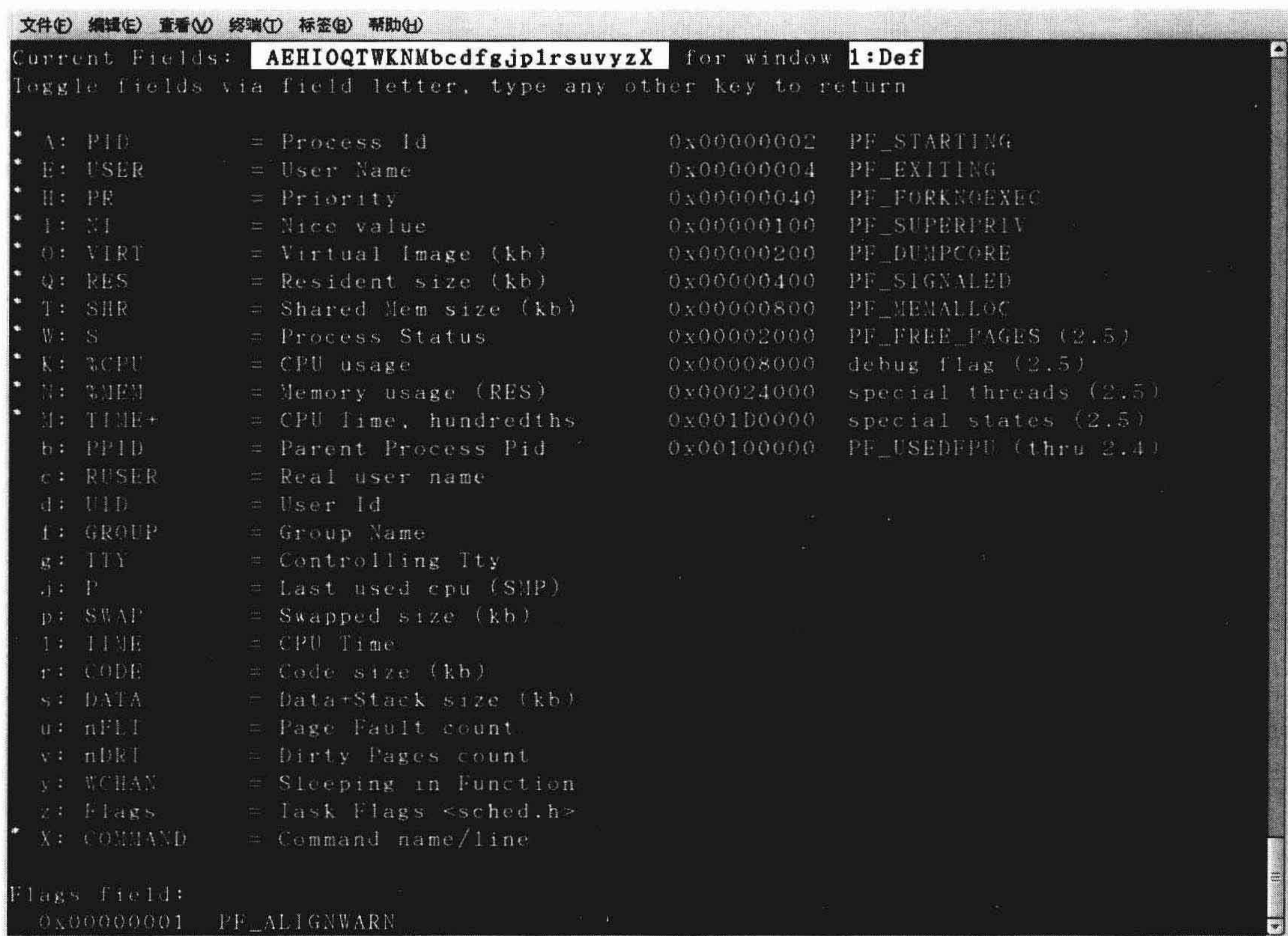


图 17-2 选择排序字段

也可以使用“<”键或“>”键快速切换排序的列，它们将分别用左边一列或右边一列进行排序。使用“R”键切换升序和降序排列。

如果想列出默认列表中没有的信息，也可以在输出列表中添加一列。按“F”键调出所有可用资源列表，如图 17-3 所示。

选定需要显示的字段后，按回车键回到任务列表。

■注：很可能因为选择的字段太多而在一屏上显示不开，此时就需要去掉一些字段来获得更多的显示空间。

在 top 命令中还有很多选项，可以在帮助菜单中获得相关信息。按“?”键可以打开帮助。如果按照喜欢的方式对 top 命令的信息显示模式进行了自定义，可以按“W”键将设置保存。设置会保存到用户 home 目录中一个叫.toprc 的文件中。

如果发现了一个运行不正常的任务，可以使用 top 命令将其退出。要退出某个程序时，按下“k”键并输入问题任务的 PID。此时，命令会询问要发送给进程的信号。使进程正常结束的信号是 15。如果 15 不管用，可以试一下更厉害一点的信号 3，或者试一下信号 9，此信号会像用斧头一样杀死进程。

更多信号，请参阅“杀并不总是杀死”的内容。

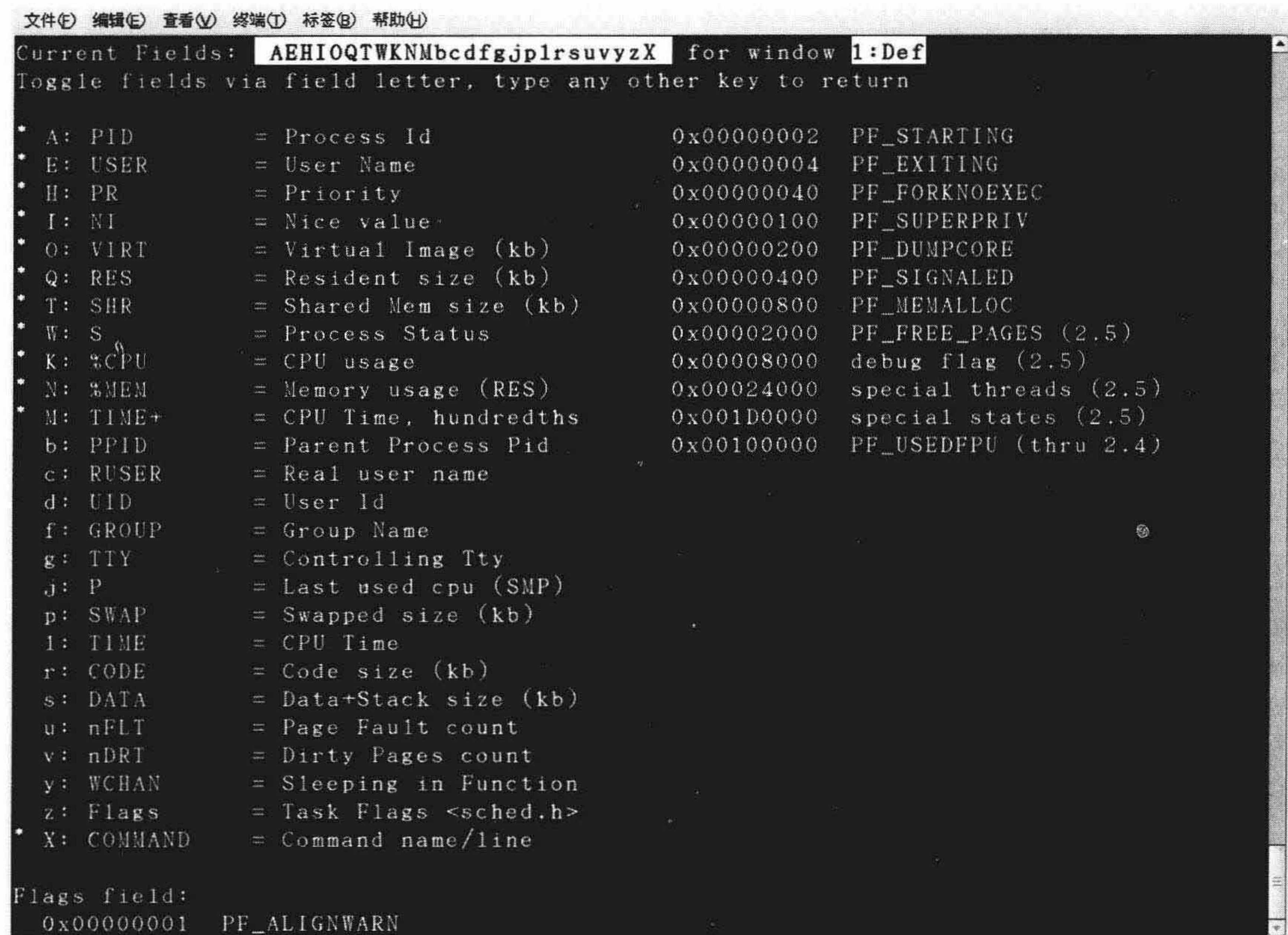


图 17-3 添加列表项

注：要向进程发送一个信号，必须是进程的拥有者或 root 用户才可以。

杀并不总是杀死

虽然向进程发送信号的操作称作 killing 并通常使用 kill 命令来完成，但此操作并不一定会使进程退出。此操作使用内核的一个功能向进程发送信号。

系统提供的服务被设计用来监听此信号，并在接收到此信号后执行一定的动作。最常用的是信号 1，也称为 HUP 信号。此信号将会使多数服务重新载入其配置文件，而不用停止和重启。

如果想退出一个应用程序或进程，可以向其发送 TERM 信号或 QUIT 信号，分别对应着 15 号和 3 号信号。

如果一个任务忽视这种礼貌的停止要求，那就需要使用信号 9 或 kill 来强迫它这样做了。可以在 man 7 signal 的手册页找到更多有关信号和运行过程的相关内容。

如果一个进程使用了太多的 CPU 时间但又不能杀死，可以试着提高它的友好度，从而降低其优先权。这就意味着此进程会更容易地向其他进程出让 CPU 的使用权，从而允许其他进程使用一些本来要分配给它的 CPU 周期。可以通过按下“R”键改变进程的友好度。按键后 top 命令会要求指定需要改变的进程的 PID 以及新的友好度值。

除非用 root 用户或通过 `sodu` 来运行 `top` 命令，否则只能提高进程的友好度。也就是说只能降低一个进程的优先权，而不能提高。

修改完后，按“Q”键退出。

■注：也可以在不使用 `top` 工具时使用 `renice` 工具改变友好度和优先权。例如，要修改 PID 为 8612 的进程的友好度，可以运行“`renice +5 8612`。”

Gnome 系统监控

当在 Red Hat 或 Ubuntu 中有 Gnome 运行时，也可以使用图形化工具显示进程和系统信息。此工具叫做 `gnome-system-monitor`，可以从“系统”>“管理”>“系统监视器”启动它，如图 17-4 所示。

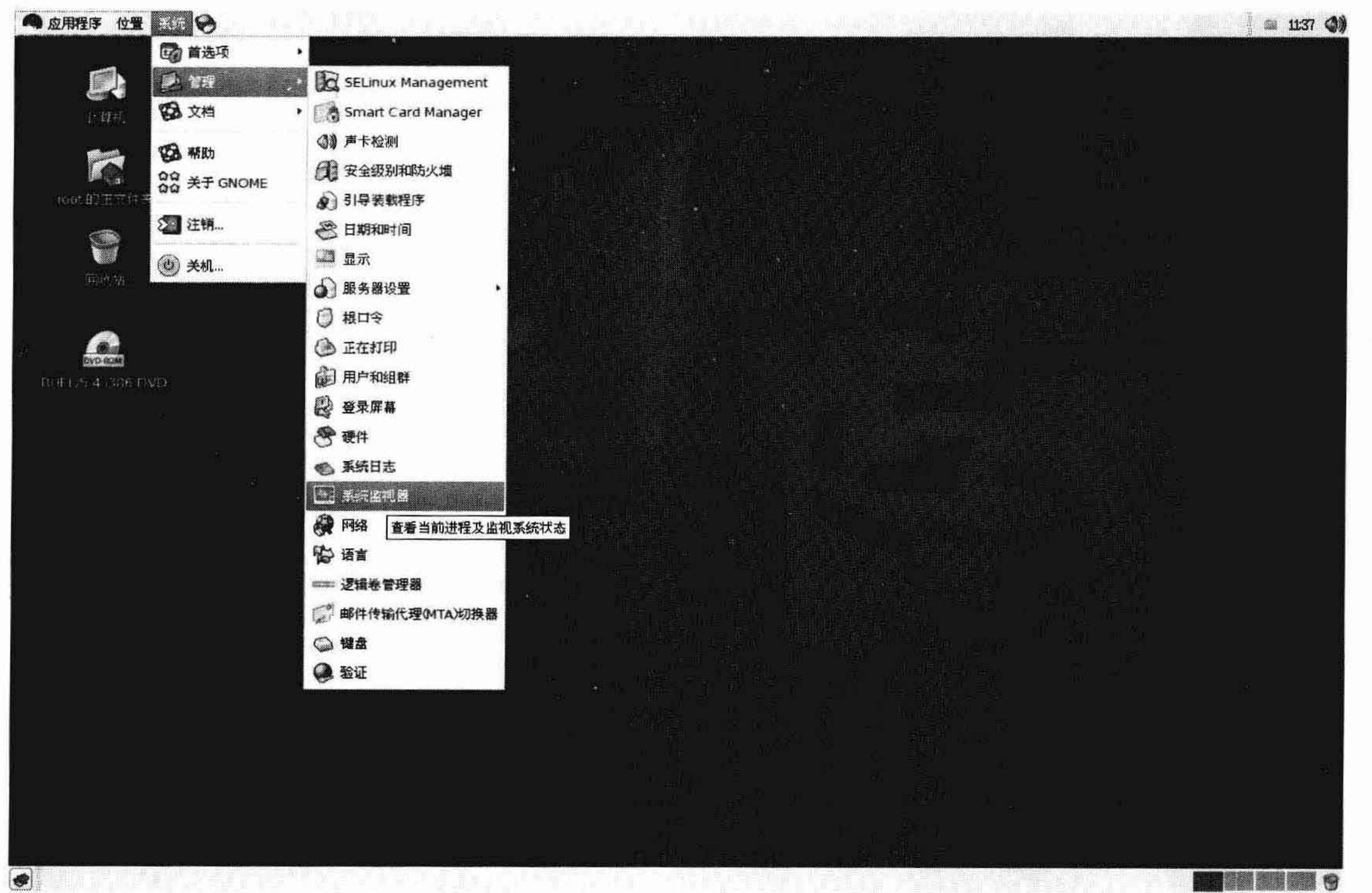


图 17-4 启动 `gnome-system-monitor`

此工具看起来很像 windows 中的任务管理器，它也提供了很多和 `top` 命令中一样的功能。它启动后默认显示的是“资源”选项卡，如图 17-5 所示。

此页面显示的内容和 `top` 工具显示的标题行中的内容一样，只不过这里还以图形的形式显示了过去 1 分钟左右的使用率信息。这就使系统是否运行在特定资源过低的状态情况更加一目了然。

要查看正在运行的任务列表，可以转到进程选项卡，如图 17-6 所示。

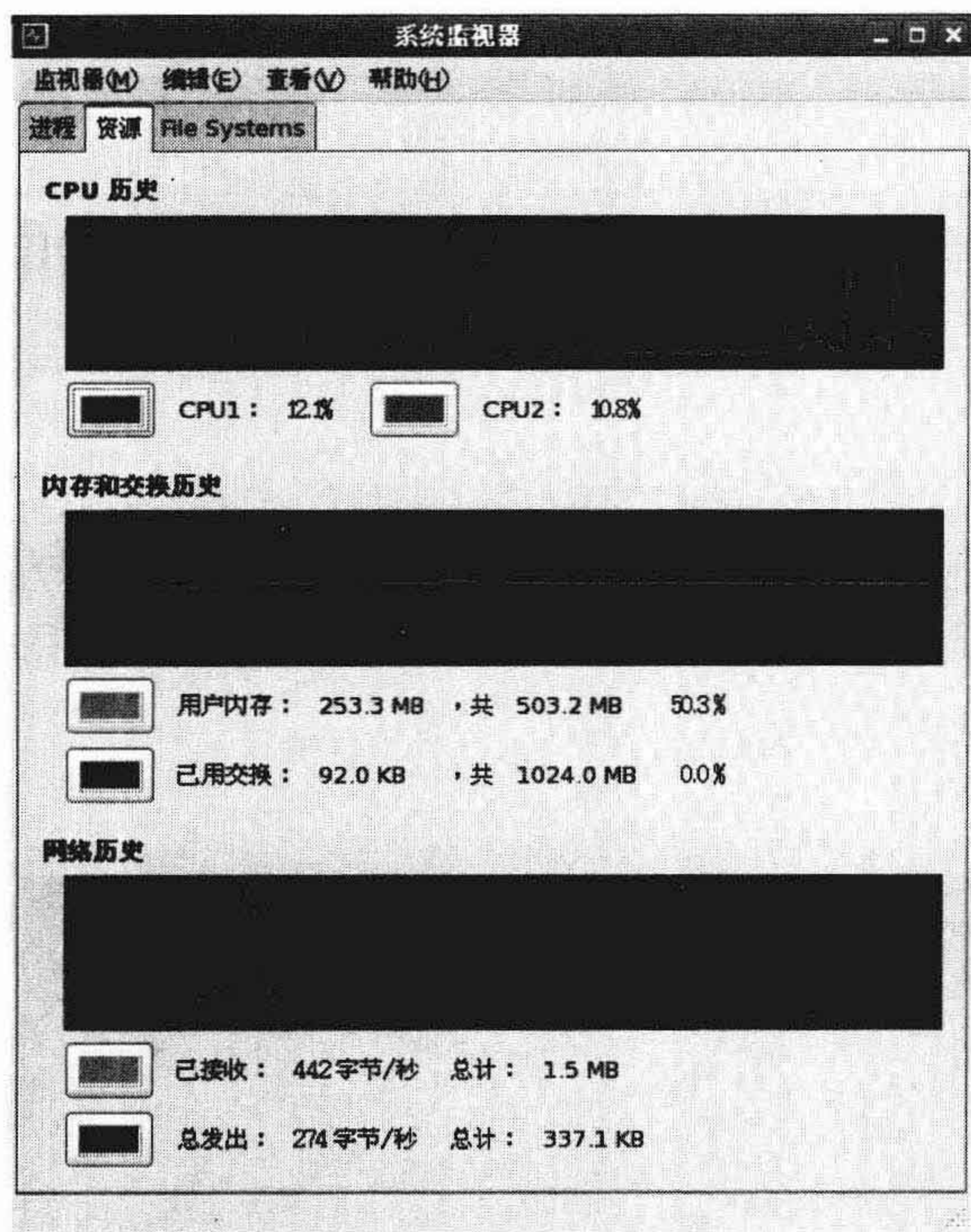


图 17-5 Gnome 系统监视器的“资源”选项卡



图 17-6 Gnome 系统监视器的“进程”选项卡

此选项卡显示了正在运行的进程的列表。可以通过单击一个特定行的标题使信息按不同列进行排序。不是所有的列都默认显示，可以添加或删除一个列。使用“编辑”➤“性能”菜单打开“系统监视器首选项”对话框，如图 17-7 所示。

要让一个字段显示在列表中，只需在进程字段列表中选中相应的复选框就可以了。此对话框中，还可以设置进程列表刷新时间间隔以及在要杀死一个进程时系统监视器是否做出警告。设置允许杀死进程时显示警告信息是一个明智的选择，因为这样设置有助于防止对进程的误杀。

在“资源”选项卡可以改变图表的刷新时间以及背景和网格的颜色。最后，在“File Systems”选项卡中可以设定文件系统信息的刷新时间以及是否将类似/proc 和/sys 之类的特殊文件系统包含在列表中。设置完成后，关闭“系统监视器首选项”对话框。

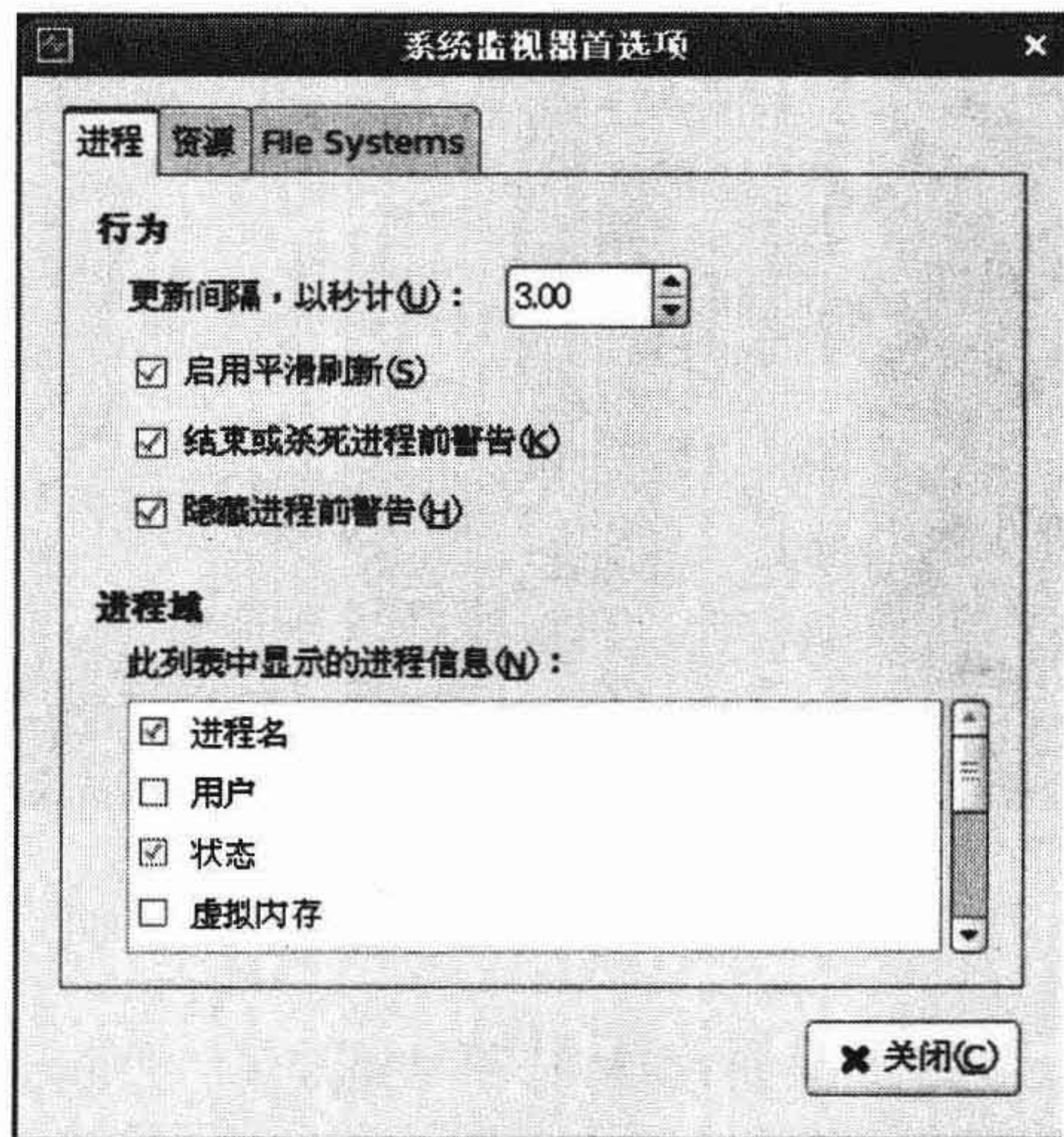


图 17-7 “系统监视器首选项”设置对话框

只有用户账号所拥有的进程会被显示在进程列表中，也可以通过选中“查看”➤“全部进程”菜单项切换到让 Gnome 系统监视器显示系统中的所有进程，如图 17-8 所示。

最后，如果要停止任何出错进程，可以从“编辑”菜单中选择“结束进程”或“杀死进

程”菜单项，也可以右键单击列表中的进程，如图 17-9 所示。



图 17-8 改变显示选项



图 17-9 停止一个进程

从弹出菜单中选中停止或杀死，此时会弹出一个确认对话框，如图 17-10 所示。



图 17-10 确认进程终止操作

■注：除非使用 root 用户运行 Gnome 系统监视器，否则将不能停止自己拥有的进程以外的任何进程。

在弹出菜单中或“编辑”菜单下有一个选项可以列出所有已经打开的文件。选中此项后，将会列出目前已打开和被进程使用中的文件，这和在第 8 章中讨论过的 lsof 命令的输出有些类似。

17.2.2 交换空间的使用

交换空间使用太多可能就显示系统内存过低，可以使用 vmstat 工具查看是不是这个原因。通常还要告诉这个工具每隔几秒钟输出一次信息以便判断出任何的趋势。如列表 17-3 所示，使用“-n”选项防止输出标题行并通知命令每 5 秒钟重新显示统计信息。按 Ctrl+C 退出 vmstat 命令。

列表 17-3 vmstat 输出信息，交换空间活动状况

```
$ vmstat -n 5,
```

procs		-----memory-----				---swap--		-----io----		-system--		----cpu----			
r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa
1	0	0	61404	13016	1518468	0	0	16	10	5	2	2	1	97	0
0	0	0	60568	13040	1519388	0	0	154	87	3383	8453	6	1	91	1
0	0	0	59484	13048	1520416	0	0	214	30	3416	13056	8	3	88	2
0	0	0	53384	13060	1522088	0	0	335	1	3432	14855	10	3	86	1
1	0	0	52416	13068	1522944	0	0	159	71	3383	15334	10	3	87	0

这个相当令人印象深刻的混杂数字显示了有多少数据正在系统中移动。每一行都包括了 6 组数字集合，这些数字表示进程、内存、交换空间、输入/输出、系统以及 CPU 等的使用信息。

由于过多交换空间的使用而导致的性能减弱将显示在两组数字中：swap 和 cpu。

在显示交换内存使用量的 swap 组中的 si 和 so 列表示从交换空间读数据（so—swap out）和向交换空间写数据（si—swap in）的信息。在 cpu 组中的 wa 列表示 CPU 等待数据而没有处理指令的时间百分比。

如果一个主机将应用程序在交换空间中移进移出上花费大量的 CPU 周期，si 和 so 列的值将增高。实际的数字将依赖于正在发生的交换数据的大小以及交换设备的速度。wa 列也会经常显示高达 90 多的值，表示此时 CPU 正等待从交换空间中取回应用程序或数据。

可以使用 top 工具找出是否有应用程序占用了太多的 RAM，如果有就将其配置改为使用较少的内存，如此可以解决此类问题。

■注：top 工具自身的运行也是需要占用资源的，因此，如果系统没有响应或出于极端的超负荷运转时，重启可能要比等待 top 工具启动要快得多。

示例中，主机没有使用任何交换空间，CPU 多数时间空闲，意味着系统运行在轻负荷状态。

17.2.3 磁盘存取

vmstat 工具也会给出有多少数据正从磁盘读出或向磁盘写入的信息。在列表 17-4 中高亮显示了 bi（块写入）和 bo（块读出）列。同时通过附加了一个秒数作为参数，使得 vmstat 会在 5 秒后自动退出。

列表 17-4 vmstat 输出信息，I/O 活动状况

```
$ vmstat 5 5
procs -----memory----- ---swap-- -----io----- -system-- ----cpu----
 r b  swpd  free   buff  cache   si   so    bi   bo   in   cs   us   sy   id   wa
 1 0    0 61404 13016 1518468   0    0   16   10    5    2    2    1   97   0
 0 0    0 60568 13040 1519388   0    0  154   87 3383 8453    6    1   91   1
 0 0    0 59484 13048 1520416   0    0  214   30 3416 13056    8    3   88   2
 0 0    0 53384 13060 1522088   0    0  335    1 3432 14855   10    3   86   1
 1 0    0 52416 13068 1522944   0    0  159   71 3383 15334   10    3   87   0
```

这两列数据准确的显示了每一段间隔时间中从磁盘读出（bi）和向磁盘写入（bo）的数据块数量。示例中，系统从磁盘中读取的数据要比写入的多，但两个数据都不高。因为磁盘数据块的大小是 4KB，所以从这里可以看出系统运行负荷不高。

什么操作会造成系统沉重的负荷取决于数据块的大小和磁盘的速度。如果正在写入一个大文件，这个数字在写操作期间会升高，但写完后就又会降下来。当在一个闲置的系统中创建一个文件时，通过检查此数字，就可以知道此主机的处理能力了。在运行 vmstat 命令的同时，运行 dd 命令在后台创建一个全 0 的 1GB 大小的文件，如列表 17-5 所示。

列表 17-5 检查磁盘 I/O 性能

```
$ dd if=/dev/zero of=./largefile bs=1M count=1024 &
[1] 21835
$ vmstat 5 5
procs -----memory----- ---swap-- -----io----- -system-- ----cpu----
 r b  swpd  free   buff  cache   si   so    bi   bo   in   cs   us   sy   id   wa
 1 1   5140 96056  96392 1598024   0    0    4    13    31   10    4    1   95   0
 2 0   5156 96512  54536 1655212   0    0    0 76164  737  3203    3   18   46   33
 0 5   5168 101772 53736 1667252   0    0    0 73152  717  3054    3   18   35   44
 0 0   5168 129984 51520 1677972   0    0   15 24598  532  1793    2    7   59   33
 0 0   5168 129984 51528 1677976   0    0    0   22   295  1196    2    0   98   0
1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB) copied, 13.1767 s, 81.5 MB/s
[1]+  Done                  dd if=/dev/zero of=./largefile bs=1M count=1024
$ rm ./largefile
```

刚开始运行时，因为文件数据还在内核缓存中，bo 列的数字依然很低。然而在下一个时间间隔，可以看到它飙升到了 76000 个数据块。但要记住，这是在 5 秒钟时间里发生的，由此可以计算出此特定主机的数据峰值数据写入率为大约每秒钟 15000 个数据块。

如果发现系统性能下降，并且使用 vmstat 测试很长时间都显示 bo 值接近系统处理能力的极值，那就一定有一个应用程序或服务正在拼命写入大量数据，这种情况通常都表示出现

了问题，因此应该找出哪些应用程序是罪魁祸首并纠正问题。

在第 18 章讨论 Nagios 时还会返回来讨论如何能够在系统性能下降时能获得警告信息。

17.3 持续性能监控

现在有了一些可以诊断性能瓶颈位置的基础工具了，下面将讲述如何使系统自动运行监控。它将给出系统长期性能表现和资源利用率的数据，利用这些数据，可以更好地确定什么时候以及如何对硬件进行升级或将服务移植到其他主机上去。

17.3.1 SNMP

持续性能监控使用一个叫做 SNMP（简单网络管理协议）的协议进行。通常一个被监视的主机上会运行一个 SNMP 服务器，也称为 **agent**（代理），它在持续运行的基础上不断搜集主机性能的统计信息。此代理可以被客户端查询，客户端再对查询到的性能数据进行收集、处理以及显示。

下面就从安装并配置 SNMP 客户端和工具开始。在 Red Hat 上 SNMP 及其工具由 `net-snmp` 和 `net-snmp-utils` 软件包提供。

```
$ sudo yum install net-snmp net-snmp-utils
```

Ubuntu 中由 `snmpd` 和 `snmp` 软件包提供。

```
$ sudo apt-get install snmpd snmp
```

安装完成后，需要更改配置文件以便于访问更多的系统性能信息。一个 SNMP 代理有 3 个资源访问等级，而且访问的等级由用来连接的社区名称决定。此社区名称有效地作为用户名和密码存在，因此选择一个不易被猜到的名字是非常明智的，否则系统信息就可能会被第三方所窥知。

首先确定代理只监听本地 Internet 网络接口。在 Ubuntu 中，这是默认的设置，但在 Red Hat 中则需要重新设置定义在 `/etc/init.d/snmpd` 中的参数的值。可以通过新建一个叫做 `/etc/snmp/snmpd.options` 的文件并在其中添加所需参数的方法来达到这个目的。这个文件将会被启动脚本引用，而其中的设置也将覆盖默认的设置。

只需要从 `init` 脚本中复制默认选项到新建的文件中并添加需要服务器监听的地址就可以了，示例中监听的地址是 127.0.0.1，如列表 17-6 所示。

列表 17-6 Red Hat 中的 `/etc/snmp/snmpd.options`

```
# Override the default daemon options.
#
OPTIONS="-Lsd -Lf /dev/null -p /var/run/snmpd.pid -a 127.0.0.1"
```

Ubuntu 中，定义这些选项的是 `/etc/default/snmpd` 文件中的 `SNMPDOPTS` 变量。

■警告：需要确保面向 Internet 的机器有防火墙进行保护，同时 SNMP 服务器不必监听外部接口上的请求。SNMP 协议使用 UDP 端口 161。

接下来，需要配置要使用的社区名称并授权其对所有系统变量的只读访问权限。此项配置在 Ubuntu 和 Red Hat 中都在/etc/snmp/snmpd.conf 文件中。

首先，必须定义一个社区名称并将其映射到一个内部的安全的名称。对示例而言，Ubuntu 有一个非常适合的配置，因此只需简单改动一下以适合示例所用就可以了。随后也可以把此配置应用于 Red Hat 主机。

列表 17-7 包括了将社区名称映射到内部名称的配置指令。

列表 17-7 默认社区映射

```
# First, map the community name (COMMUNITY) into a security name
# (local and mynetwork, depending on where the request is coming
# from):
#          sec.name source      community
com2sec    paranoid default    public
#com2sec    readonly default    public
#com2sec    readwrite default   private
```

由于不需要 paranoid 类型的名称，因此可以将其注释掉。在这里使用具有 readonly 安全属性的名称，因为已经配置过这个属性的名称可以访问所有有效信息。把具有此属性的条目前的注释符去掉，加上安全属性，最后修改一下相关的社区名称。这里使用 axs4snmp 作为社区名称。列表 17-8 是修改过的配置片段。

列表 17-8 自定义的社区映射

```
# First, map the community name (COMMUNITY) into a security name
# (local and mynetwork, depending on where the request is coming
# from):
#          sec.name      source      community
#com2sec    paranoid     default     public
com2sec    readonly     default     axs4snmp
#com2sec    readwrite    default     private
```

Ubuntu 中的用户组映射不需要修改，但是还是在列表 17-9 中列了出来，以便模仿此配置并应用在 Red Hat 中。由于具有 patanoid 和 readwrite 安全属性的用户在配置中没有定义，因此也不必费心对其相应的用户组映射进行修改了。这里用到的用户组只有 MyROGroup。

列表 17-9 默认用户组映射

```
# Second, map the security names into group names:
#          sec.model sec.name
group MyROSystem v1      paranoid
group MyROSystem v2c     paranoid
group MyROSystem usm     paranoid
group MyROGroup v1      readonly
group MyROGroup v2c     readonly
group MyROGroup usm     readonly
```



```
group MyRWGroup v1      readwrite
group MyRWGroup v2c      readwrite
group MyRWGroup usm      readwrite
```

其中由 SNMP 代理提供的主机信息并不是在 `snmpd` 自身中定义的，而是来自管理信息库 (MIB) 的。这些管理信息库描述了主机的可用信息并以树结构的形式显示出来。每一条可检索的（或可设置的，如果代理配置允许这样做）信息都称为一个对象标识符 (OID)。`snmpd` 使用的管理信息库定义在 `/usr/share/snmp` 中。

■注: 更多 MIB 和 OID 相关信息, 请参阅“http://en.wikipedia.org/wiki/Simple_Network_Management_Protocol#Management_Information_Bases_.28MIBs.29”。

客户端可以检索的主机信息数量和类型由一个 `view` 来定义。每一个 `view` 定义都包括 `view` 名称和可以访问的 OID 树分支。列表 17-10 定义了一个可以对 “.1” 下所有可用信息进行访问的叫做 `all` 的 `view`，还定义了一个可以对 “.iso.org.dod.internet.mgmt.mib-2.system” 下所有信息进行访问的叫做 `system` 的 `view`。

列表 17-10 定义 view

```
# Third, create a view for us to let the groups have rights to:
#           incl/excl subtree      mask
view all    included .1            80
view system included .iso.org.dod.internet.mgmt.mib-2.system
```

接下来，需要定义访问等级，用来指定哪一个用户组访问哪一个 `view`。列表 17-11 显示了 Ubuntu 中的默认配置，此默认配置可以用到示例中。此处定义了 `MyROGroup` 用户组对所有的 `view` 都有只读访问权限。

列表 17-11 默认访问权限配置

```
# Finally, grant the 2 groups access to the 1 view with different
# write permissions:
#       context sec.model sec.level match read write notif
access MyROSystem ""    any noauth exact system none none
access MyROGroup ""    any noauth exact all none none
access MyRWGroup ""     any noauth exact all all none
```

上述就是为了能从代理处检索主机性能和统计信息而对配置文件所做的仅有的改变。当然，还可以定义一些主机的位置和联系信息以便于稍后对其进行查询时更容易识别。

定义上述信息，可以设置第三个文件 `/etc/snmp/snmpd.local.conf` 中的 `syslocation` 和 `syscontact` 变量，如列表 17-12 所示。

列表 17-12 snmpd.local.conf

```
# Set host specific information
syslocation Melbourne, Australia
syscontact Hostmaster <hostmaster@example.com>
```

好了，现在可以重启 SNMP 服务器了，在 Red Hat 中使用 “`sudo service snmpd`” 命令，在 Ubuntu 中使用 “`sudo invoke-rc.d snmpd`” 命令。

■提示：SNMP 配置可以有点黑色艺术。比如，要快速启用 snmpd，可以用命令“echo 'rocommunity xs4snmp default' | sudo tee /etc/snmp/snmpd.conf” 创建单个只读的社区，此操作将会覆盖已有的配置文件。

Red Hat 和 Ubuntu 都带有一个叫做 snmpconf 的 SNMP 配置向导工具。使用此向导，只需简单地回答几个关于主机的问题就可以自动生成一个配置文件。如果读者感觉上述内容中讨论的配置并不适用于自己的主机，就可以使用这个向导工具创建一个自定义的配置。

17.3.2 Cacti

现在主机中有 SNMP 代理来收集信息了，但是却并没有利用这些统计信息做什么事情。本节将讨论如何通过 Cacti 来利用这些信息。Cacti 是一个基于 Web 的应用程序，它可以从各种来源收集数据并生成图表，从而使用户更容易地发现趋势。

示例中将使用一个名为 cacti.example.com 的主机访问统计信息。首先，在 DNS 服务器上创建一个 CNAME 或一个 A 入口项并准备好 Apache 所需的配置指令和文件。但是，在文件系统中的/srv/www 下创建目录时先不要创建最终的 html 目录。确认在 Red Hat 的默认主页列表中添加了 index.php 文件名。

■注：在第 9 章讨论过如何配置 DNS，第 11 章讨论过如何创建虚拟 Web 主机。

1. 在 Ubuntu 中安装

Ubuntu 的软件包系统中已经有 Cacti 了，所以可以使用命令“sudo apt -get install cacti”来进行安装。当询问使用什么样的 Web 服务器时，选择“None”，如图 17-11 所示。

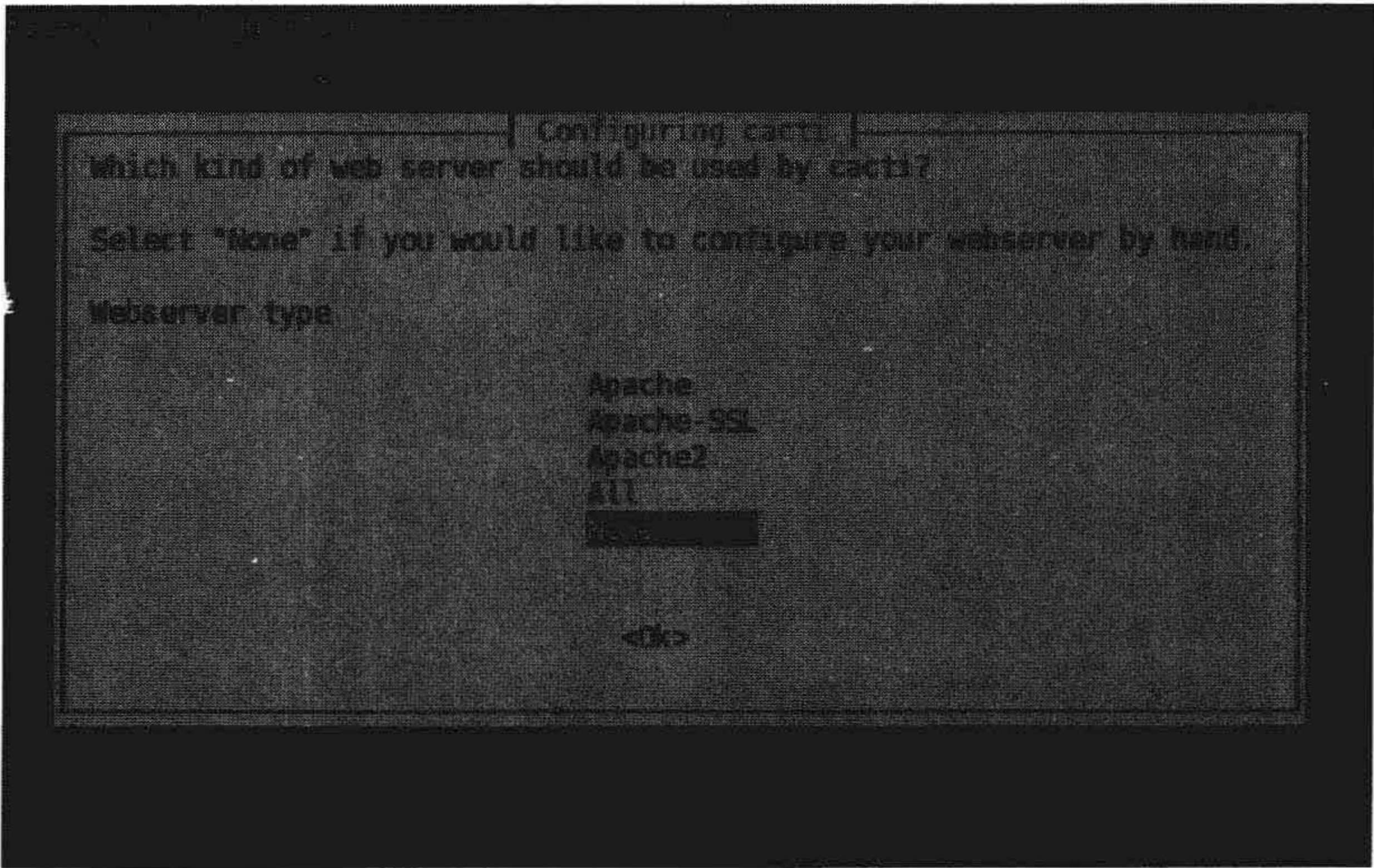


图 17-11 Cacti 配置

接下来，软件包会自动安装。安装完成后，还会询问一些用来配置数据库的信息，Cacti 将数据存储在 MySQL 数据库中。

使用 `dbconfig-common` 框架来管理此配置意味着这些设置将会在更新事件中被自动应用，因此选择 “Yes”，如图 17-12 所示。

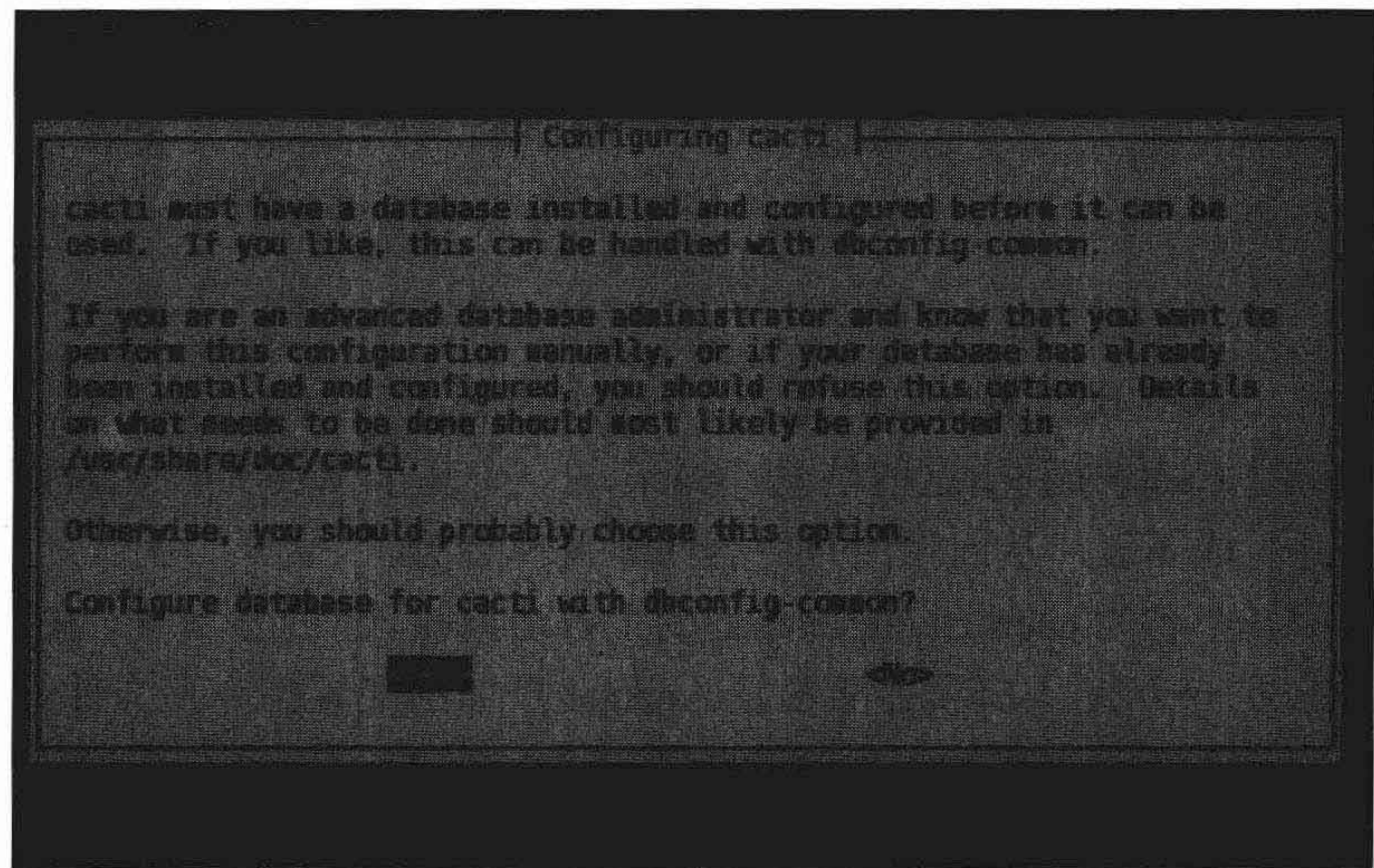


图 17-12 指定使用 `dbconfig-common`

接下来会要求填写 MySQL 数据库 `root` 用户的用户名和密码以便于为 Cacti 创建一个 MySQL 账号。然后还要为 Cacti 账号输入一个密码。这里将密码框留空，系统会自动产生一个安全密码，如图 17-13 所示。

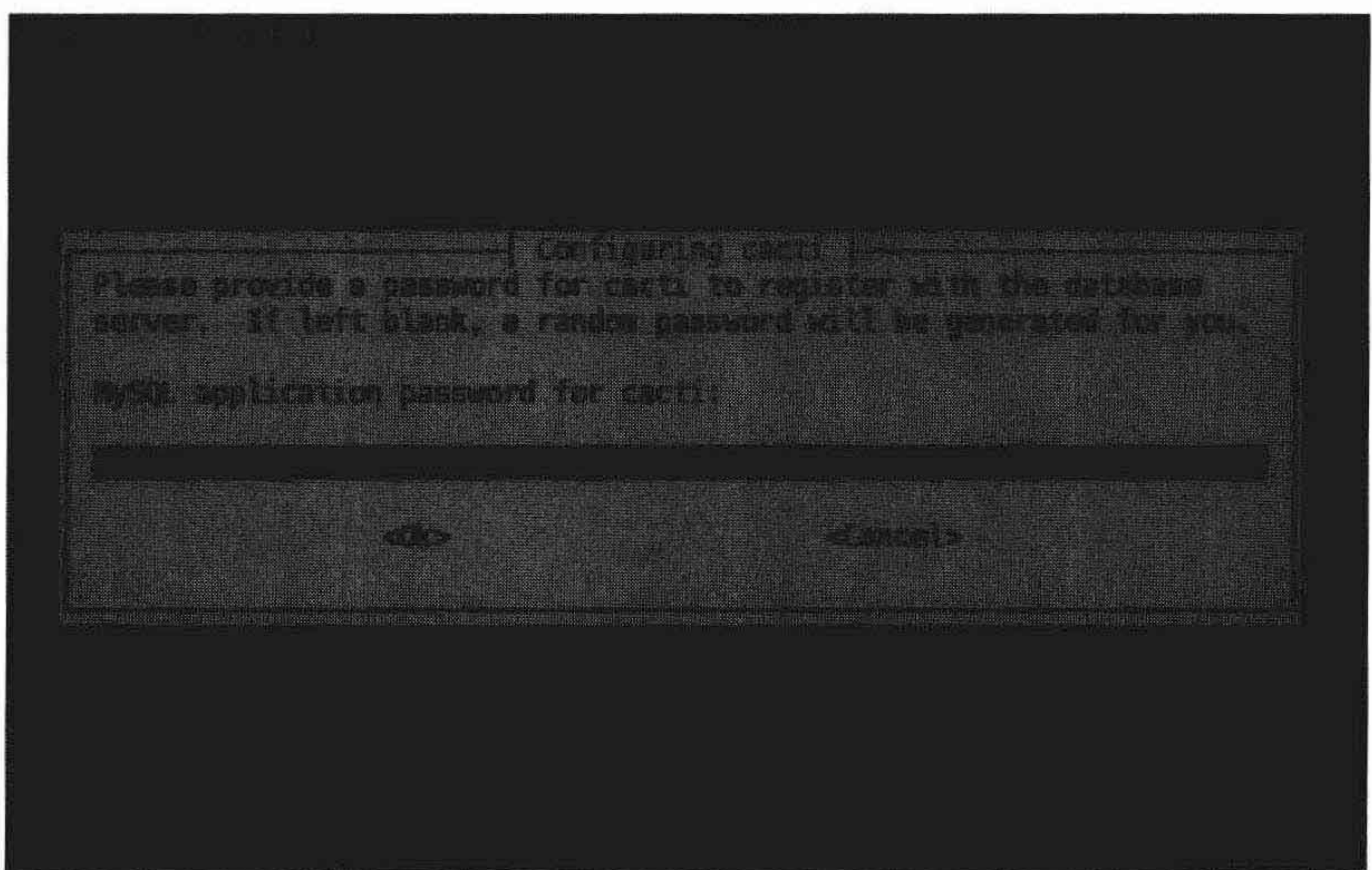


图 17-13 产生密码

按下回车键接受此默认设置后，系统将产生一个 MySQL 数据库用户，创建一个名为

cacti 的数据库并在库中加入一些表。

现在可以将虚拟主机中的 `html` 目录改为指向由软件包系统安装的 `cacti` 版本处了，可以使用符号链接实现此操作。此例中，运行如下命令。

```
$ sudo ln -s /usr/share/cacti/site /srv/www/cacti.example.com/html
```

因为 `Apache` 允许跟踪符号链接，因此当访问虚拟主机时，服务器将会转向 `Cacti` 打包的版本。

■提示：Ubuntu 中很多打包的 Web 应用程序将其站点内容安装在 `/usr/share` 目录下。可以使用命令“`dpkg -L 软件包名`”找到软件包的具体安装位置，然后就可以在自己的虚拟主机配置中使用了。

2. 在 Red Hat 中安装

在 Red Hat 中，`Cacti` 就不是现成的了，不过添加了对 PHP 的 SNMP 支持后，可以从 Fedora 的 EPEL 库中下载并安装所需的软件包。

```
$ sudo yum install php-snmp
$ sudo rpm -Uvh ➡
http://download.fedora.redhat.com/pub/epel/5/i386/rrdtool-1.2.27-3.el5.i386.rpm ➡
http://download.fedora.redhat.com/pub/epel/5/i386/cacti-0.8.7b-1.el5.noarch.rpm
```

或者可以先把 EPEL 库安装在主机上，然后再从已安装的库中安装所需的软件包。如果 EPEL 库没有添加到 Yum 配置文件中，可以通过安装 `epel-release` 的 RPM 包来添加 EPEL 库。

```
$ sudo rpm -Uvh ➡
http://download.fedora.redhat.com/pub/epel/5/i386/epel-release-5-3.noarch.rpm
```

然后就可以使用 `yum` 安装软件包了。

```
$ sudo yum install cacti
```

■注：在 RPMForge 中也有 `Cacti` 的 RPM 软件包。

`Cacti` 软件包将会安装一个 `Apache` 配置片段，此配置片段将会在每一个虚拟主机上创建一个 `/cacti` 子目录。示例中不需要这样的设置，所以通过重命名的手段将其屏蔽。

```
$ sudo mv /etc/http/conf.d/cacti.conf /etc/http/conf.d/cacti.conf.disabled
```

在可以服务于 `Cacti` 软件包的内容之前，还需要改变一下 SELinux 上下文以使 `Apache` 可以获得访问授权。

```
$ sudo chcon -R httpd_sys_content_t /usr/share/cacti
```

然后就可以将虚拟主机上的 `html` 目录链接到软件包系统所安装的 `Cacti` 版本处了。

```
$ sudo ln -s /usr/share/cacti /srv/www/cacti.example.com/html
```


因为 Apache 允许跟踪符号链接，所以当访问虚拟主机时，服务器将会转向 Cacti 打包的版本。

■提示：Red Hat 中很多打包的 Web 应用程序将其站点内容安装在/usr/share 目录下。可以使用命令“rpm -ql 软件包名”找到软件包的具体安装位置，然后就可以在自己的虚拟主机配置中使用了。

最后，因为 Cacti 使用 MySQL 数据库存储数据，还需要创建一个 MySQL 用户及数据库并将详情输入到配置文件中。

```
mysql> CREATE DATABASE `cacti`;
Query OK, 1 row affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON cacti.* TO `cacti`@`localhost`
IDENTIFIED BY 'secret';
Query OK, 0 rows affected (0.12 sec)
```

现在可以把这些配置值填写到 Cacti 的配置文件/etc/cacti/db.php 中去了，如列表 17-13 所示。

列表 17-13 Cacti 配置文件

```
/* make sure these values reflect your actual database/host/user/password */
$database_type = "mysql";
$database_default = "cacti";
$database_hostname = "localhost";
$database_username = "cacti";
$database_password = "secret";
$database_port = "3306";
```

最后还要允许 Cacti 轮询，这是一个使用 cron 每 5 分钟从所有配置过的主机中收集一次信息的小工具。软件包已经安装了/etc/cron.d/cacti，所要做的只是把命令前的注释符去掉，如列表 17-14 所示。

列表 17-14 Cacti 轮询 cron 入口

```
*/5 * * * * cacti /usr/bin/php /usr/share/cacti/poller.php > /dev/null 2>&1
```

3. Cacti 配置

可以通过访问新的虚拟主机来完成安装。访问时，将会被转到/install/子目录下，在这里可以看到安装和升级的相关信息以及 GPL 许可。如果同意此许可的内容，直接单击“Next”来开始安装。

在下一页中，Cacti 显示了前面输入的数据库设置并提供了两种安装选择，一种是是否要运行一个新的安装，另一种是是否要对已安装的版本进行升级，如图 17-14 所示。

下一页，如图 17-15 所示，可以看到 Cacti 已经自动检测到了所有收集数据和产生图表所需的命令行工具。如果愿意，也可以手动更改这些路径，不过对示例而言，默认的设置就可以了。

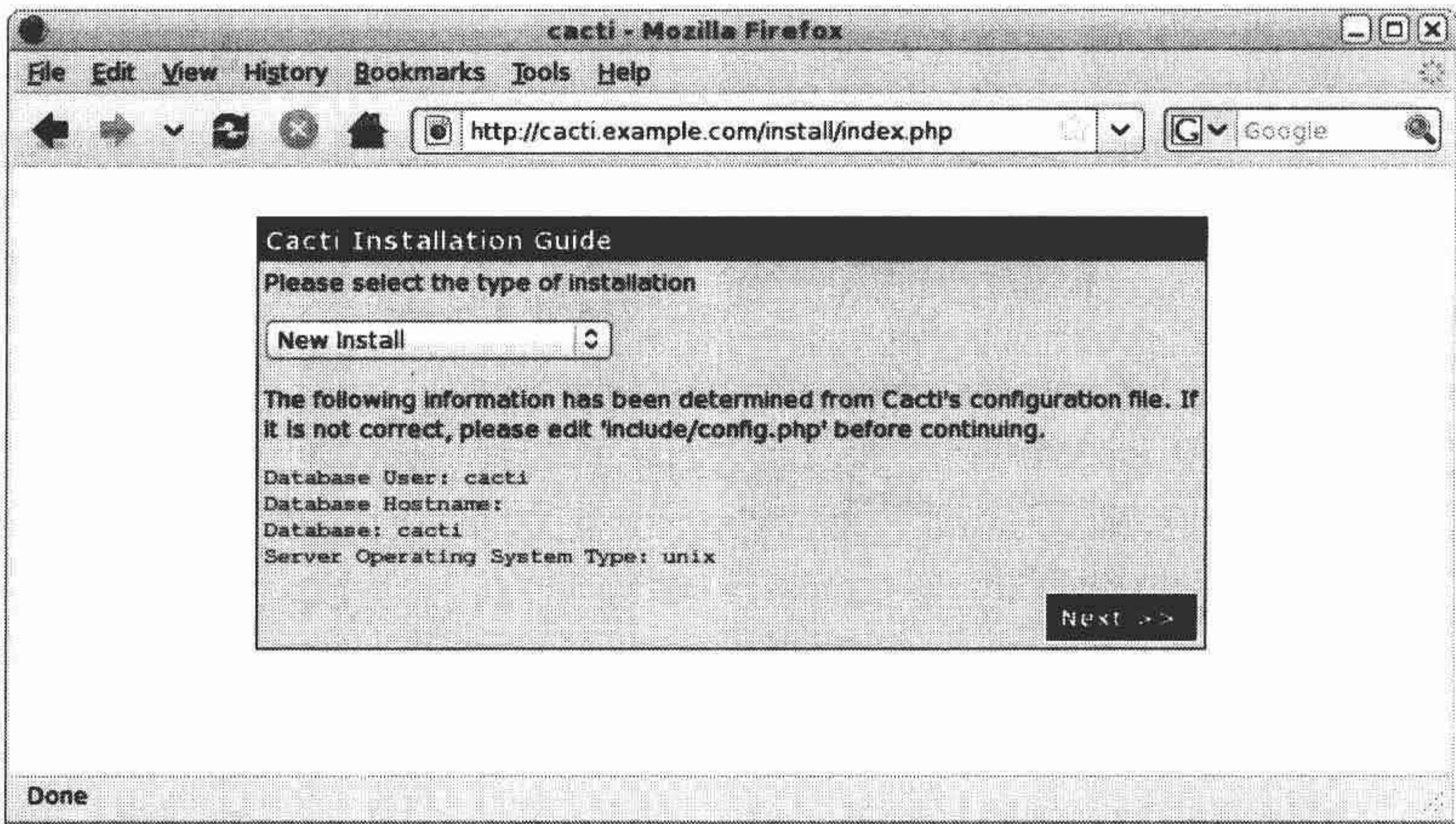


图 17-14 选择安装类型

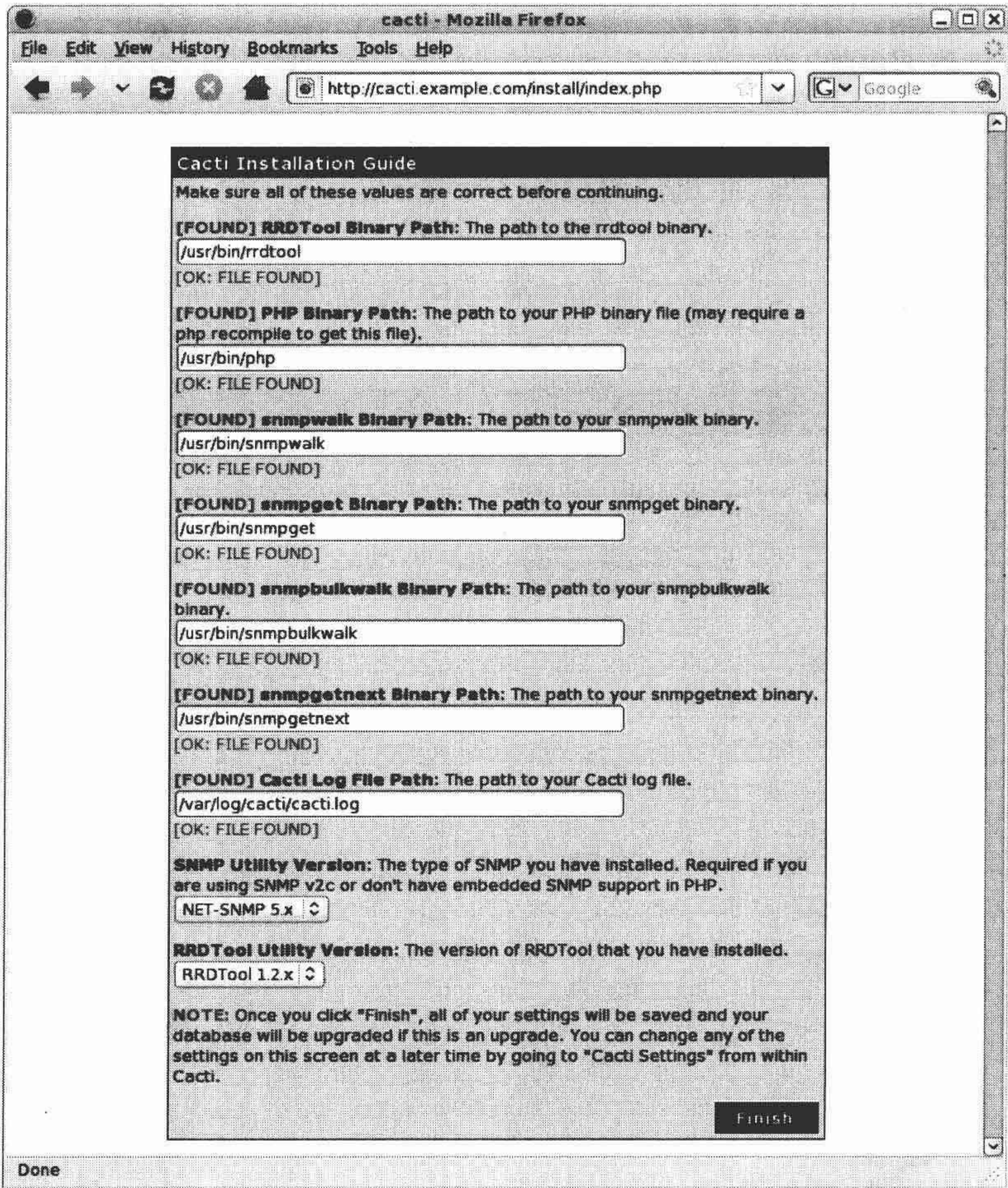


图 17-15 应用程序定位助手

单击“Finish”按钮接受已经设置的所有值并登录 Cacti。安装完成后，产生了一个名为 admin 的用户，如图 17-16 所示，此用户的默认密码也是 admin。

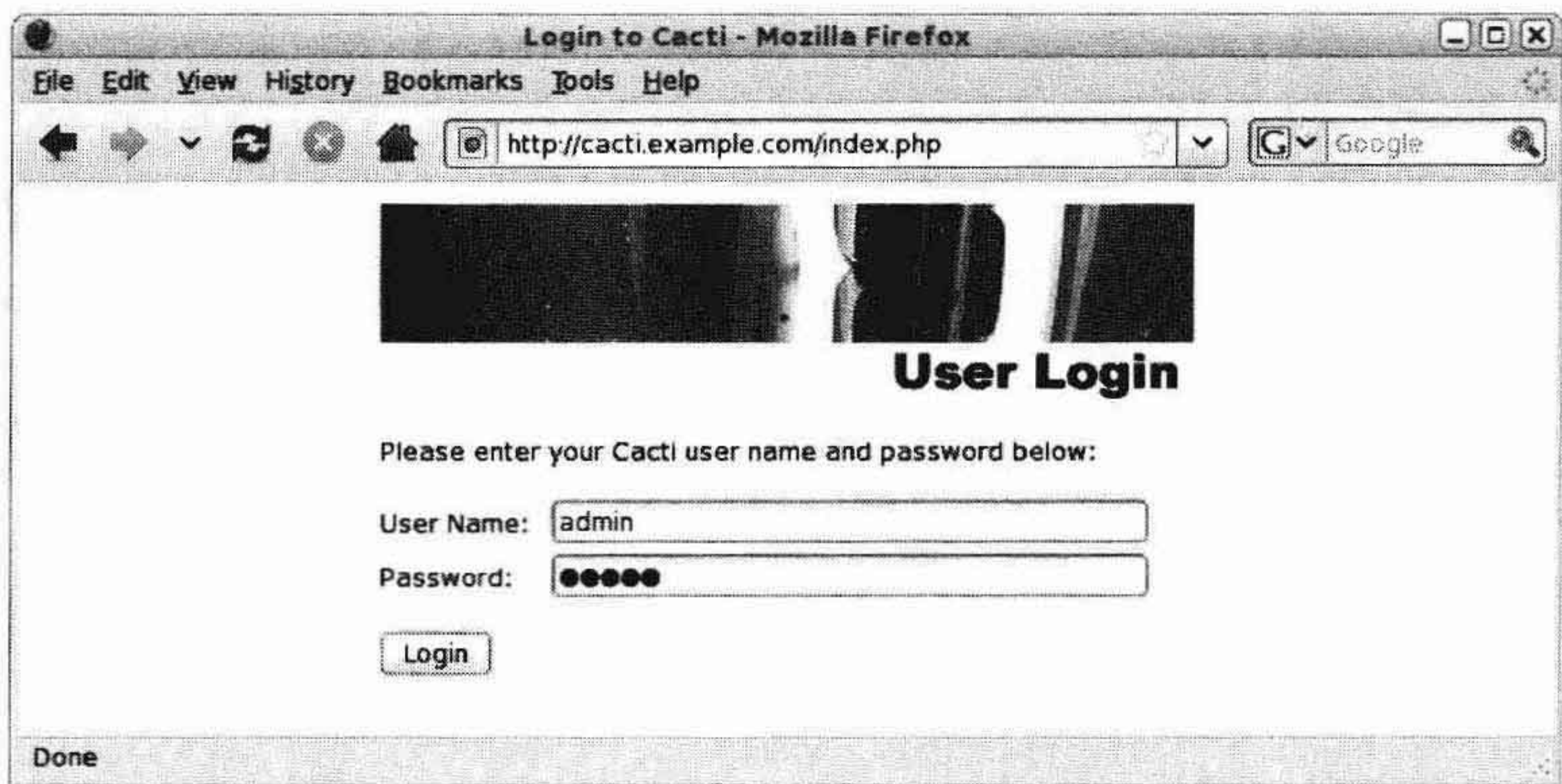


图 17-16 首次登录 Cacti

首次登录后，将会提示修改默认密码。选一个安全的不易被猜中的密码填上。如果第三方获得了 Cacti 安装的访问权，入侵者将会获得关于网络和主机的潜在有用信息。

4. 添加主机到 Cacti

现在已经登录了 Cacti，就可以将代理或主机的信息告知它，然后它就可以收集和图表化数据了。用 Cacti 里的术语说，主机称为设备（device）。

在主控制台中，如图 17-17 所示，开始创建设备。

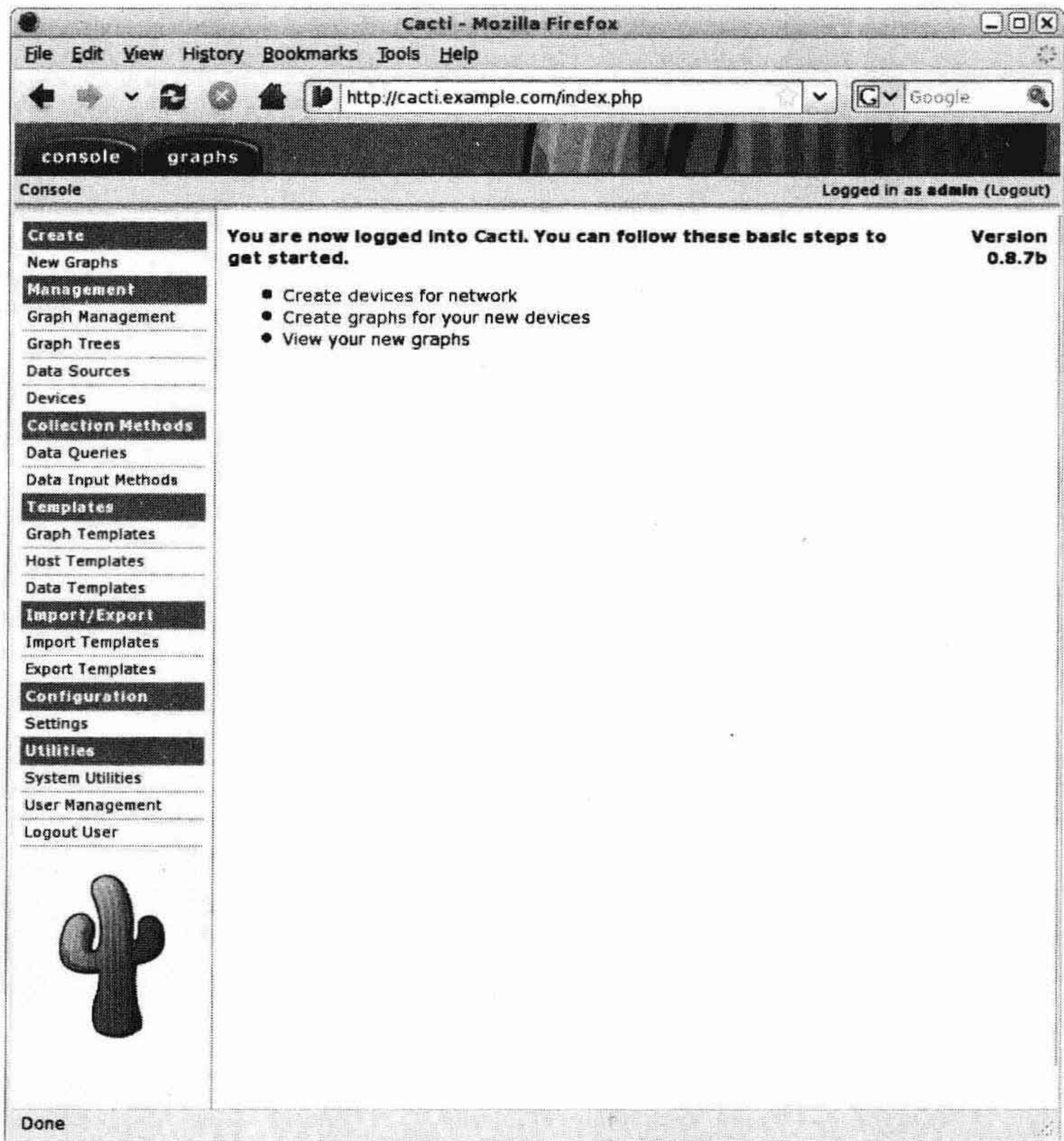


图 17-17 Cacti 的主控制台

如图 17-18 所示，安装时就把本地主机自动地添加进来并且已经开始收集数据了。在右上角单击“Add”来创建一个新设备。

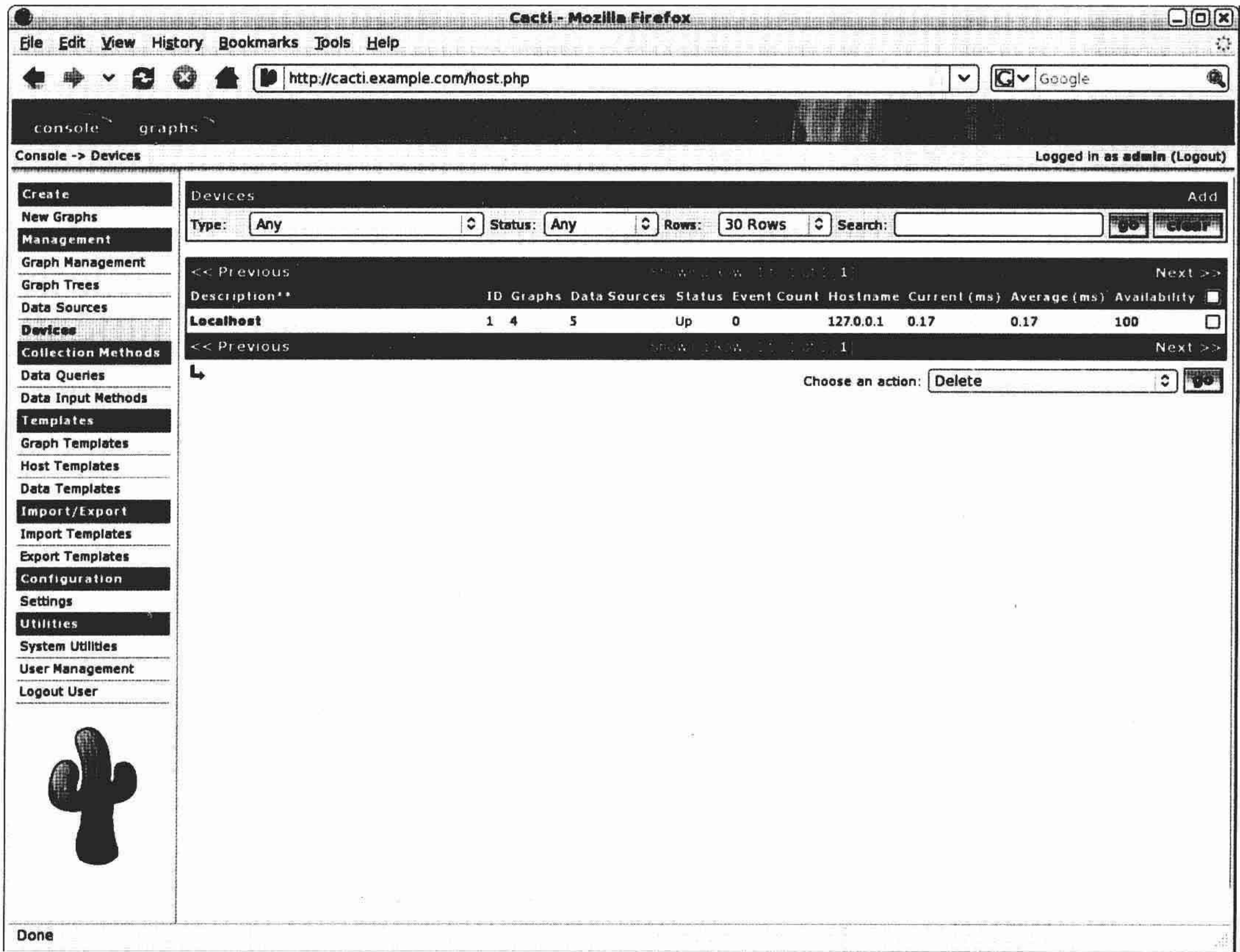


图 17-18 设备列表

在设备安装页面，如图 17-19 所示，输入要加入的设备的描述及主机名称或地址。因为将要使用 SNMP 来对主机进行查询，所以将主机模板（Host Template）设为“Generic SNMP-enabled Host”。

接下来，选择“Version 2”作为使用的 SNMP 的版本，这样一来就可以将离线设备检测（Downed Device Detection）项设为“Ping and SNMP”，并设置 ping 的类型（Ping Method）为“TCP Ping”，ping 端口（ping port）为 22。

按上述设置离线设备检测项后，如果 Cacti 不能建立一个 SNMP 链接同时也不能连接到列出的 TCP 端口时，主机将被标识为离线。在 SNMP 选项（SNMP Options）下，设置 SNMP 社区（SNMP Community）为“axs4snmp”。其他选项可以保持默认。

单击“Create”按钮向 Cacti 中添加一个设备。添加完成后将转到显示设备详细内容的页面，但是如果一切正常，会在靠近页面顶部的位置输出从 SNMP 代理处获取的信息，如图 17-20 所示。

值得注意的是，列出的正常运行时间是 SNMP 代理的正常运行时间，而不是主机的正常运行时间。

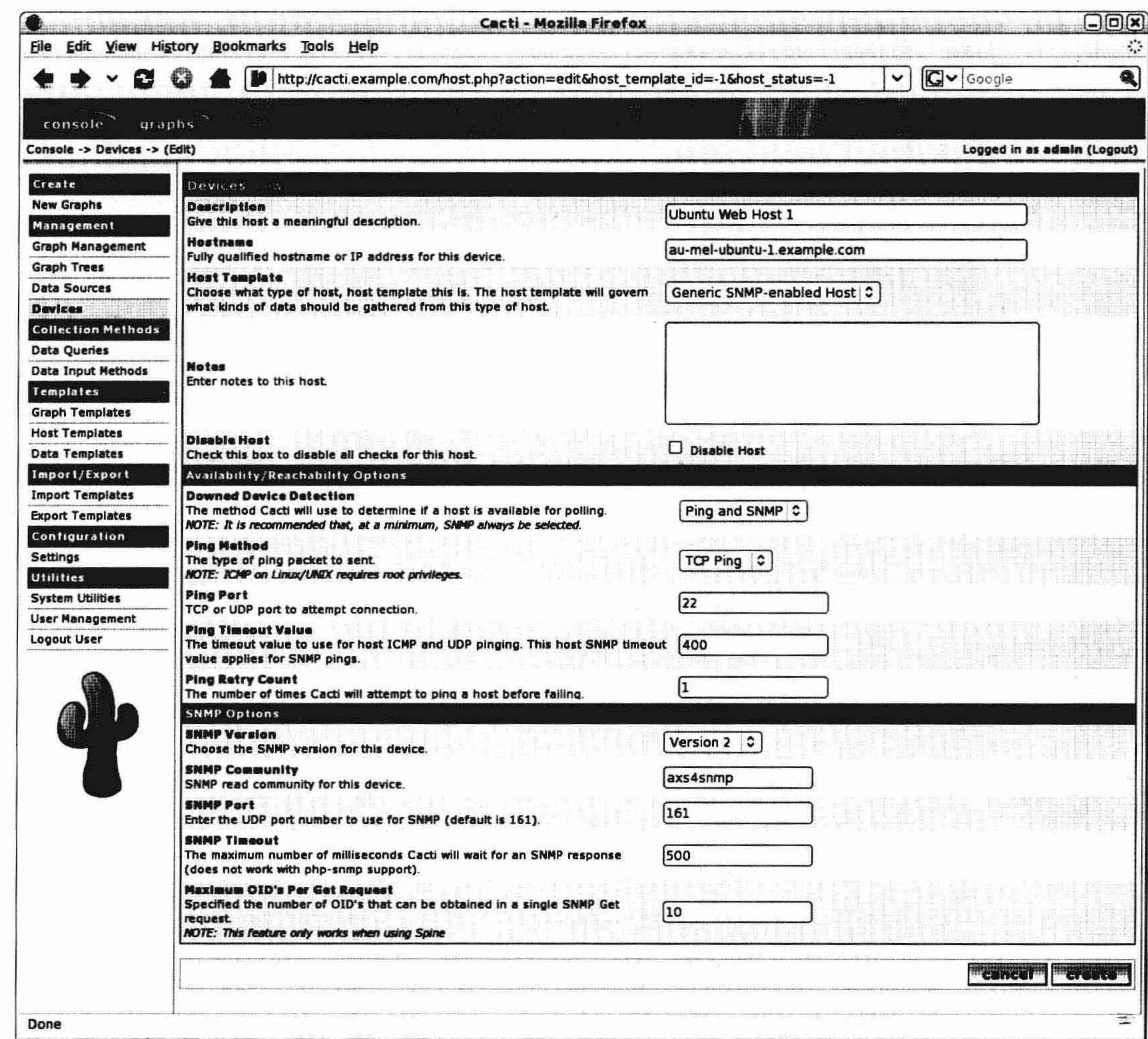


图 17-19 添加新设备

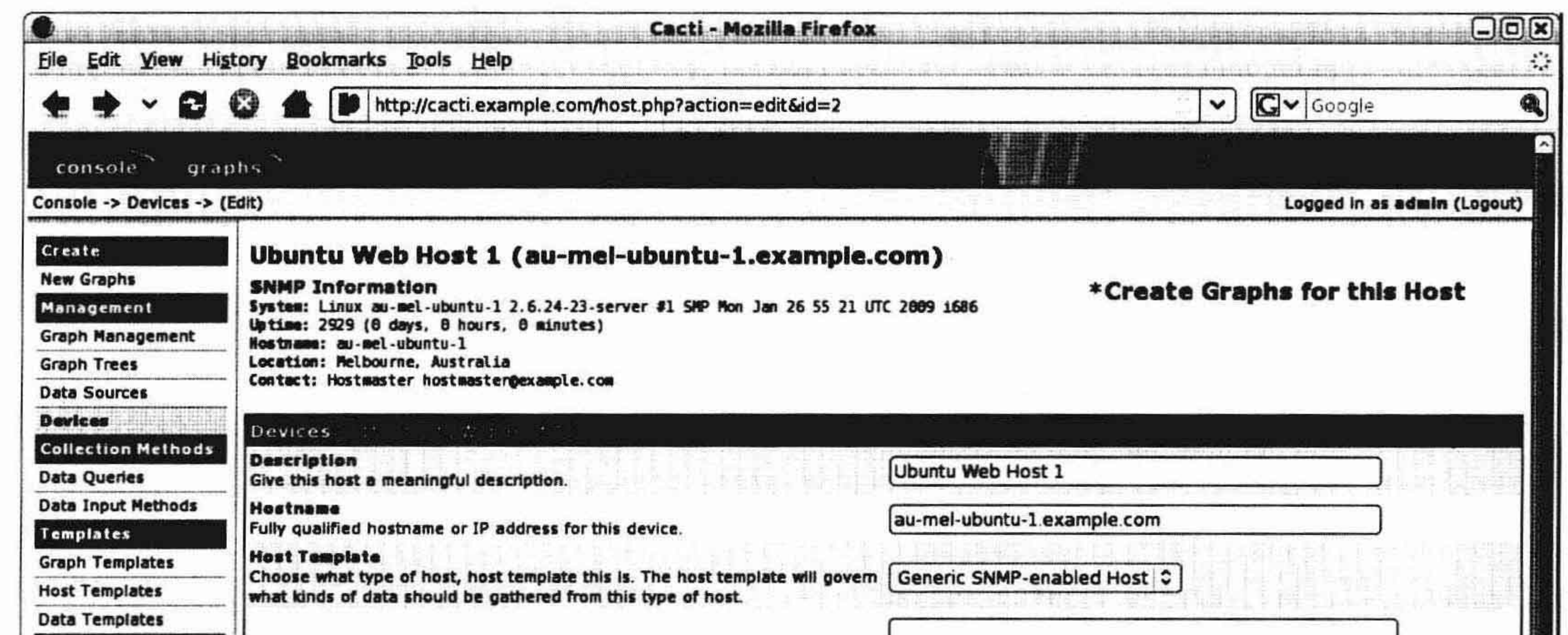


图 17-20 设备添加成功

提示：如果遇到 SNMP 错误，请检查 Cacti 主机是否可以连接远程的 SNMP 代理。例如，可以使用 snmpwalk 工具并提供协议版本、社区字符串以及主机名等选项来检查，示例中可以使用以下命令 “snmpwalk -v2c -c axs4snmp au-mel-ubuntu-1.example.com”。

现在 Cacti 可以从代理处收集数据了，但是还要告诉它哪些数据要生成图表以及想使用这些数据生成哪种图表。

单击 “Create Graphs for this Host”（为主机创建图表），在下一页选中行 eth0 接口后面的复选框。图表类型设置为 “In/out Bytes with Total Bandwidth”（总带宽下输入/输出字节），如图 17-21 所示，然后单击 “Create” 按钮。

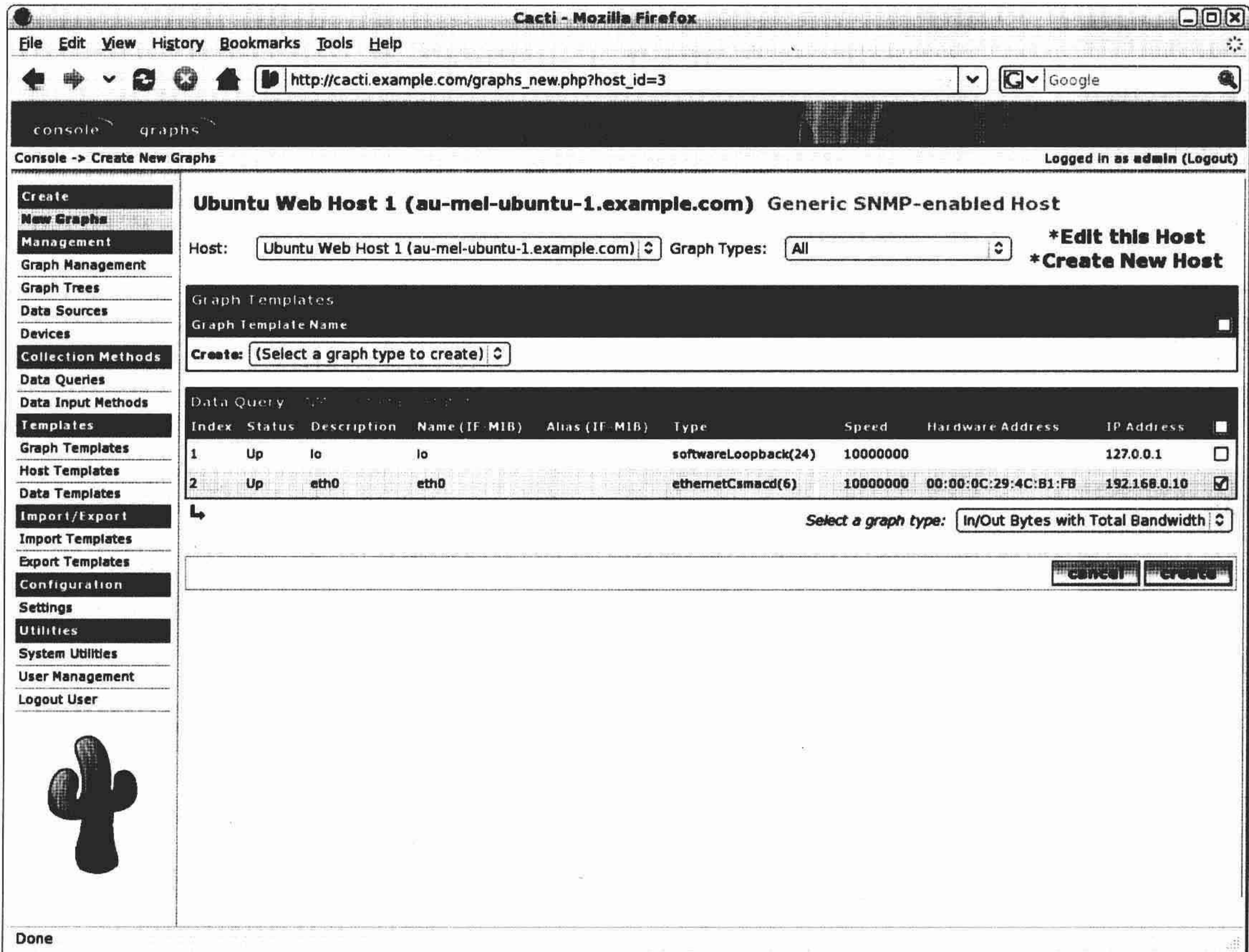


图 17-21 创建新图表

同样的页面将被重新载入，但是 eth0 行以灰色显示，表示此图表已经存在。在可以查看此图表之前，需要将其加入一个图表树。一个图表树就是一个主机或图表的简单的层次结构，在此树中以类型或主机为标准对图表进行分组显示。

从左侧的菜单中选择 “Graph Trees”（图表树），单击默认树，单击 “Add”。此页中，选择要添加的节点在树中的显示位置。这里把节点加入到 [root] 中，当然也可以添加在任何不同的树或标题下。要添加的树节点是一个主机，名称为 “Ubuntu Web Host 1”，

如图 17-22 所示。“Graph Grouping Style”（图表分组类型）默认即可。完成后，单击“Create”按钮。

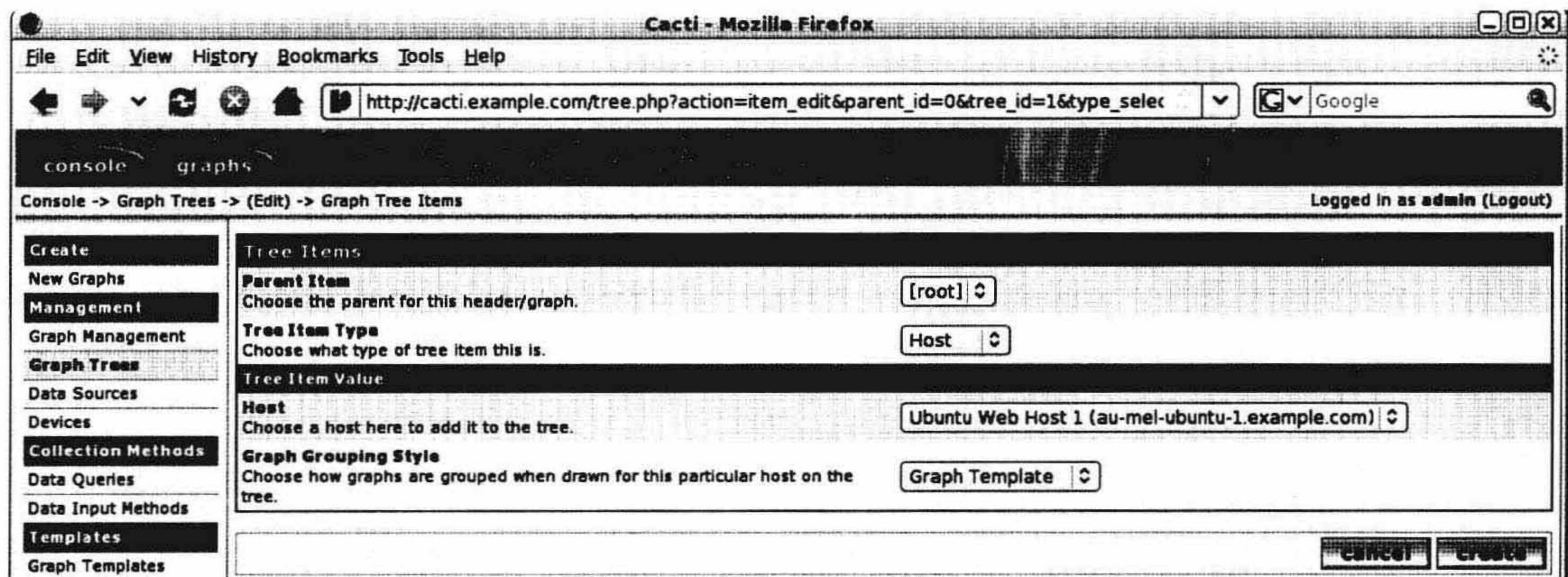


图 17-22 在图表树中添加一个节点

这样就添加了一个图表，下面就可以查看它了。单击页面顶部大大的蓝色图表选项卡并在树中选择“Ubuntu Web Host 1”。现在就可以欣赏到初始的网络流量图了，如图 17-23 所示。

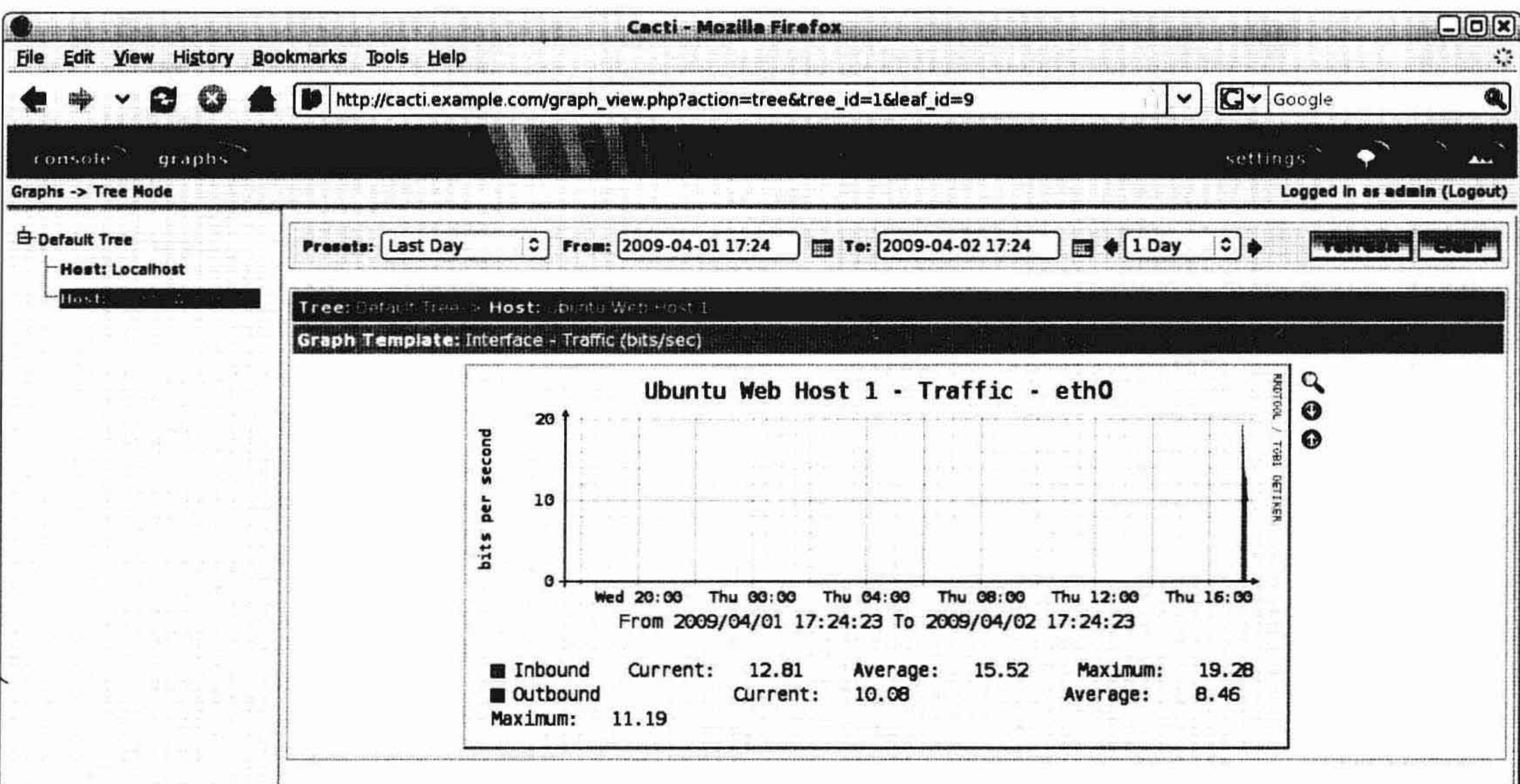


图 17-23 新的网络流量图

如果单击图表图像本身，将会转到一个显示了 4 个图表的页面，每一个都有不同的时间间隔。这就可以进行日常的监控，也可以进行长期的监控了。

当单击其中一个图表旁边的放大镜时，鼠标的光标将会发生变化，这时可以选中图表中的一个区域。完成选择后，图形将放大到只显示所选区域部分。

要添加更多的图表，单击蓝色的控制台选项卡，在左侧的菜单中紧跟着的是设备项，然

后单击“Ubuntu Web Host 1”，紧跟着的是为这个主机创建图表。

从图表模板列表中选择“Host MIB-Logged in Users”（主机 MIB—已登录的用户），然后单击“Create”。程序要求为图表图例选择一种颜色，如图 17-24 所示。

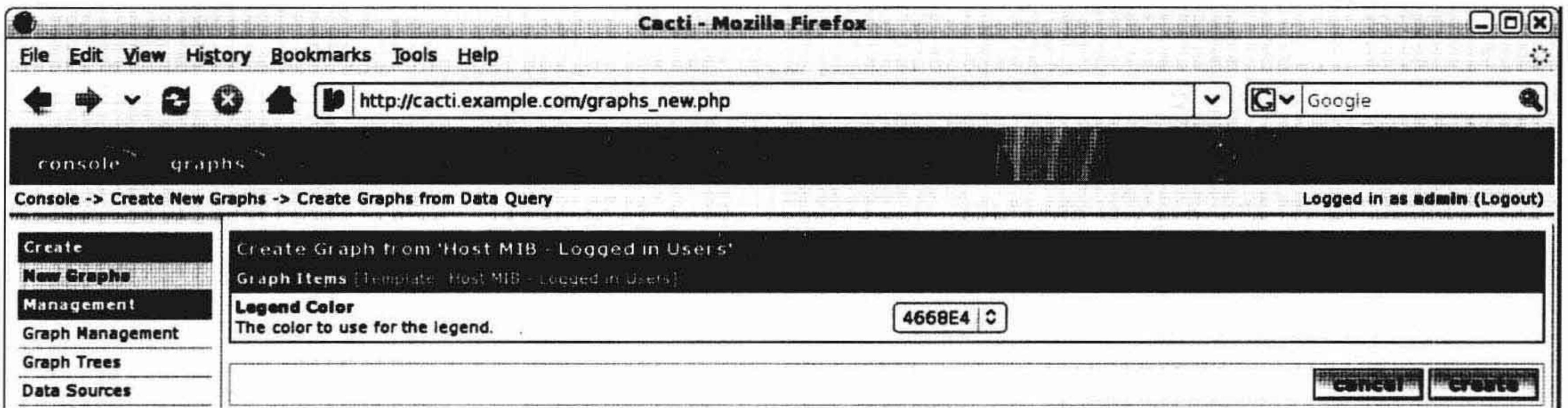


图 17-24 选择颜色

如果不喜欢默认选项，可以选择一个新的颜色，然后单击“Create”按钮。完成后将返回到图形创建页面，但此时在页面顶部会有一个状态信息，提示新的图表已经创建完毕。

因为已经将整个主机添加到了图表树中，所以，任何添加到主机中的新的图标也会被添加到图表树中。

能实时监控主机上的进程数量、CPU 和内存的利用率以及平均负荷量也是不错的选择，因此也将这些图表添加到主机中。如果跳回到图表列表页面，将会看到刚刚创建的图表看起来一塌糊涂，这是因为轮询还没来得及收集数据。最多再过 10 分钟（那时轮询将已经返回两次信息了）就会有数据了，图表也就会被整齐地创建了。

5. 设置与默认

不必为每一个新添加的设备修改设备离线检测方式和 SNMP 社区名，只需使用左侧菜单“配置”下的“设置”链接对默认的设置进行修改就可以了。

在通用选项卡中，可以自定义默认的日志和 SNMP 设置。修改默认配置的版本为 Version 2 并将 SNMP 社区名称设置为前面选定的名称，然后单击保存按钮存储新的设置。在路径选项卡中可以修改首次配置 Cacti 看到的帮助工具的位置。除非路径发生了改变，否则此页面内容无需更改。

接下来是轮询选项卡。在此选项卡中告诉 Cacti 使用已经配置过的哪个轮询进行数据收集。在示例中，使用 cmd.php 和每 5 分钟运行一次的 cron 进行轮询。如果要改变时间间隔，也可以在这里进行设置。还可以在此选项卡中修改设备离线检测方式，这里将其设为“PING and SNMP”，使用“TCP ping”方法，端口为 22。

■注：如果 Cacti 需要访问远程的 SNMP 代理，则需要设置此代理监听一个可以访问的网络接口，同时主机的防火墙应该通过类似下面的规则允许访问“-A Firewall-eth0-INPUT -p udp -m state --state NEW -s 192.168.0.1 --dport 161 -j ACCEPT”。

在图表输出选项卡中可以设置 Cacti 将生成的图标发送到一个远程的目录或主机中，这就意味着可以将这些图表包含在一个外部可访问的系统状态页面上，比如一个无需授权就可以从外部访问 Cacti 自身的页面。

如果要改变 Cacti 显示图表的方式或图表自身的字体大小，可以在视觉选项卡中进行设置。在最后一个选项卡——认证选项卡中，可以配置外部认证源使用 Cacti，比如 Apache 的 httpd 认证或外部的 LDAP 服务器，当然使 LDAP 服务器使用 Cacti 的前提是已经安装了 PHP 的 LDAP 支持。

如果选择继续使用内置的认证，可以使用工具菜单中的用户管理页面对用户进行管理。要创建一个新用户，单击右上角的添加链接并选择一个用户名和密码。在页面下半部分的复选框使管理员可以对每个用户单独进行授权。

上面讨论了如何建立 Cacti 来进行基本的监控，但同时也可以设计用来下载并安装额外的图表模板，以检测额外的设备和服务。如果有一个设备没有任何一个模板适合它，也可以从 Cacti 内部为它定义一个新的模板。

可以在链接“<http://www.cacti.net/>”处找到更多关于 Cacti 的内容，在“http://cacti.net/additional_scripts.php”处和 Cacti 论坛“<http://forums.cacti.net>”中有更多的数据收集脚本和模板。

■注：其他备选的监控解决方案有 MRTG（<http://oss.oetiker.ch/mrtg>）和 Munin（<http://munin.projects.linpro.no/>）。

17.4 性能优化

主机安装完成后，就已经设置了大多数能够提供合理性能的默认配置了。然而，主机各有不同，因此，常常可以针对具体的工作量调整设置，使配置更加优化。

本章中要介绍的一些小提示和小技巧，将会有助于提高主机上大多数服务器相关任务的运行速度。

为了速度重新编译系统？

可能有人 would 认为对内核进行重新编译并禁用某些不需要的功能是提高主机运行速度的好方法。尽管这种方法会提高系统重启的速度，但是却不会改变内核的运行速度，因为尽管去掉的代码不再运行，但不管怎样这些代码还是在内核中存在的。

当然，重新编译过的内核使用的内存会略少一点，但是完全可以通过将可选的驱动编译成模块形式来防止其自动加载，而不用重新编译内核。比如，如果不需要 IPv6 网络协议，可以在 `/etc/modprobe.d/blacklist` 文件中加入“`blacklist ipv6`”来阻止 ipv6 内核模块的运行。

对应用程序来讲，也是同样的道理。禁用功能是会一定程度上减少内存占用，但通常不会有明显的速度提升。对于像 Apache 或 PHP 之类的应用程序，可以通过非常简单地禁用一些无用的模块来减少内存的使用量。记住，与其花几个小时的时间调整主机的配置，还不如多买块内存来得划算。

无论如何都要重新编译内核？

如果无论如何都决定要重新编译内核，第一步就是获得源码。这个随时都可以从 <http://www.kernel.org> 上下载到最新的稳定的内核源代码压缩包。最好将源代码解压到 `/usr/src` 目录下，因为此目录是系统范围内可用的。

下一步要做的就是阅读源代码中附带的 `README` 文件，并确认系统满足要编译的内核的最低要求。还需要确认已经安装了 `CGI C` 编译器和所有必须的开发库文件，其中包括帮助配置系统内核的工具所需的库文件。下表列出了最常用的 3 种内核源代码配置方式所需要的库文件软件包。

发行版	menuconfig	xconfig	gconfig
Red Hat	ncurses-devel	qt4-devel	gtk2-devel
Ubuntu	libncurses5-dev	libqt4-dev	libgtk2.0-dev

接下来，运行自己喜欢的一种配置工具对内核源代码进行配置。在命令行环境中，使用 `menuconfig` 进行配置。在基于 `KDE` 的界面上，可以使用 `“make xconfig”`，在 `Gnome` 环境中可以使用 `“make gconfig”`。一定要保证启用了所有硬件和功能的驱动及相关选项。如果忘了其中的任何一个，内核就有可能无法启动主机。

在 `Red Hat` 上，可以使用 `“make rpm”` 命令使系统构建脚本生成一个包括新内核及模块的 `RPM` 文件。在 `Ubuntu` 中，要先安装 `fakeroot` 和 `kernel-package` 软件包。安装好后，可以使用命令 `“fakeroot make-kpkg --revision=myhostname.1 --initrd kernel-image”` 生成一个内核的 `DEB` 软件包。

上述两种方法都会生成一个稍后可以直接安装的内核软件包。

注意事项

请注意，如果发现漏洞或影响使用的缺陷时，若没有使用发行版内核，则需要自己费心再编译和安装新内核。如果安装了新的硬件，可能也需要重新配置和编译新内核，以提供适合的驱动程序。

17.4.1 资源限制

`Linux` 允许使用 `ulimit` 命令对用户资源使用进行限制。这是一个 `Bash shell` 中自带的功能。所有使用此命令设置的限制只能在当前 `shell` 及由当前 `shell` 启动的进程中起作用。

加上 `“-a”` 选项运行 `ulimit` 命令可以显示当前限制设置，如列表 17-15 所示。

列表 17-15 显示当前限制设置

```
$ ulimit -a
core file size      (blocks, -c) 0
data seg size       (kbytes, -d) unlimited
scheduling priority (-e) 0
file size            (blocks, -f) unlimited
pending signals      (-i) 16384
max locked memory    (kbytes, -l) 32
max memory size      (kbytes, -m) unlimited
open files           (-n) 1024
pipe size            (512 bytes, -p) 8
POSIX message queues (bytes, -q) 819200
real-time priority   (-r) 0
```



```
stack size          (kbytes, -s) 10240
cpu time            (seconds, -t) unlimited
max user processes          (-u) 16384
virtual memory        (kbytes, -v) unlimited
file locks            (-x) unlimited
```

为了防止使用本 shell 的用户运行 fork 炸弹，可以改变此 shell 用户运行中进程的最大数量，比如使用“-u”选项将默认的 16384 改为 1。改完后，如果试着运行一个新的进程，系统将会阻止新进程的运行。

```
$ ulimit -u 1
$ ls
bash: fork: Resource temporarily unavailable
```

系统显示出错信息，提示 shell 不能产生用户尝试运行的命令的进程。

提醒：如果用户使用 SSH 或 X Window 登录的系统，则此用户就已经运行好几个进程了。

其他比较有用的限制有最大内存使用限制，可以使用“-m”选项进行设置；还有打开文件数量限制，可以使用“-n”选项进行设置。

在 man bash 页的 ulimit 部分可以找到完整的选项和功能列表。

对所有用户设置限制

ulimit 命令只对当前用户设置限制，因此可以使用/etc/profile 或/etc/bashrc 对已登录的用户添加限制。然而，很多守护进程并不使用 shell，所以在 shell 的配置文件中添加限制对这些进程毫无作用。

可以换做使用 PAM 的 limiets 模块。当用户创建一个进程时就会激活此模块，而且此模块可以根据文件/etc/security/limits.conf 中的定义设置限制。文件中可以为用户和用户组定义软件和硬件限制。列表 17-16 中是 Red Hat 的 limits.conf 示例文件。

列表 17-16 limits.conf 示例文件

#<domain>	<type>	<item>	<value>
##	soft	core	0
##	hard	rss	10000
#@student	hard	nproc	20
#@faculty	soft	nproc	20
#@faculty	hard	nproc	50
#ftp	hard	nproc	0
#@student	-	maxlogins	4

每行指定一个限制适用的域。该域的名称可以是一个通配符、用户名或用户组名。接着是限制的类型，可以是软限制或硬限制。硬限制只能由 root 用户来设置，而且当用户或进程试图打破这一限制时，系统会进行阻止。软限制可以由用户使用 ulimit 命令进行调整，但最大只能调整到对硬限制设置的值。

接下来是被限制的资源。示例文件中完整地列出了所有可用的资源，此列表也可以在“man limits.conf”页看到。每一行的最后一个值是要设置的限制的值。

示例文件中，根据所做的软限制，faculty 用户组可以同时运行 20 个进程。同时，组中的任何成员都可以将其限制值最高设到 50，但不能超过 50。

17.4.2 sysctl 和 proc 文件系统

在第 8 章中，作为一种可以直接从内核中获取信息的方法，简单地提到过 proc 文件系统。同时它还提供了修正运行中的内核以提高系统性能的方法。为此，可以修改/proc/sys 目录下虚拟文件中的值。

依据对系统部分的影响，虚拟文件被分组存储在子目录中，如表 17-2 所示。

表 17-2 /proc/sys 子目录	
目录	使用者
abi	系统仿真（比如在 64 位主机上运行 32 位应用程序）
debug	空文件夹，未使用
dev	具体设备的信息
fs	文件系统设置于调整
kernel	内核设置于调整
net	网络设置于调整
sunrpc	Sun 远程过程调用（NFS）设置
vm	内存、缓存、快速缓存设置

在这里不会详细讨论每一个文件，但会给出各种可以进行调整的思路。

每一个虚拟文件中都包含了一个或多个可以使用 cat 或 sysctl 工具读出的值。要读出一个值，可以将其关键字作为参数传递给 sysctl 工具。此关键字就是文件的完整路径，只不过要去掉路径中的/proc/sys 并有选择地将反斜杠用句点代替。比如，要查看主机使用交换内存的情况，可以查看文件/proc/sys/vm/swappiness 中的内容，查看命令为“sysctl vm.swappiness”。

命令的返回值显示了内核有多大可能将稍后不再使用的数据从 RAM 中移动到交换空间中，数值越大，可能性越高。此值的取值范围是 0 到 100，默认为 60。如果要确保主机尽可能少地使用交换空间，可以使用 sysctl 工具和-w 选项将此值设为 20，然后还要将要修改的关键字及其新值作为参数传递给命令。

```
$ sudo sysctl -w vm.swappiness=20
vm.swappiness = 20
```

另一个例子是在系统中能同时打开的文件和目录的数量，此值定义在文件/proc/sys/fs/file-max 中，并可以使用命令“sysctl fs.file-max”读出。如果要把打开文件的最大数量增加到 50 万个，可以运行命令“sysctl -w fs.file-max=500000”。

提醒：proc 文件系统中持续变化的内核变量不仅有积极的影响，也会对系统性能产生极大的不利影响。因此，除非有足够的理由，强烈建议不要对其做任何修改。

系统重启后，这些变量会被重置为默认值。要避免其值重置，可以在文件 `/etc/sysctl.conf` 中添加合适的设置条目。这样，当系统重启时，文件中的所有设置就都可以使用 “`sysctl -p`” 命令应用到系统中了。

在 Linux 内核源代码的 `documentation/sysctl` 目录下，有可用变量的详尽列表及相关说明文件，也可以参考在线文档 “<http://www.mjmwired.net/kernel/documentation/sysctl/>”。

17.4.3 存储设备

在第 8 章磁盘出错事件中曾看到过，一旦添加了更换磁盘，内核将需要重新创建磁盘阵列，而此任务所产生的大量 I/O 操作可能会降低机器提供的其他服务的平均性能。或者，希望以牺牲服务性能为代价，而给予重建进程更高的优先级。

当重建事件发生时，内核将重建的速度保持在设定的最小值和最大值之间，这些值可以在 `/proc/sys/dev/raid` 目录中使用 `speed_min_limit` 和 `speed_max_limit` 进行修改。

较为可以接受的最小速度值可以是每磁盘每秒钟 20000KB，可以使用 `sysctl` 命令进行设置。

```
$ sudo sysctl -w dev.raid.speed_limit_min=20000
```

将最小值设置得太高会影响系统的性能，所以一定要将此值设置得比系统可以处理的最大吞吐量除以磁盘阵列中的磁盘数后的数还要低。

最大值已经足够高了，此值也可以通过设置变量 `dev.raid.speed_limit_max` 进行修改。若要使磁盘阵列重建任务对系统产生尽量小的影响，可以降低此值，当然，代价就是需要等待更长的时间。

若要永久改变此值，可以把这一对值设置在 `/etc/sysctl.conf` 文件中。

17.4.4 文件系统调整

每次访问文件或目录，哪怕只是读，也会将最近访问日期戳（或叫 `atime`）更新并写入磁盘。除非需要此时间戳，否则可以通过设置主机不更新此信息来加速磁盘访问。

要这样做，只需简单地为一个要启用此选项的文件系统在文件 `/etc/fstab` 中的选项字段添加 `noatime` 项就可以了。

```
UUID=f87a71b8-a323-4e8e-acc9-efb0758a0642 / ext3 defaults, ➡  
errors=remount-ro,noatime 0 1
```

运行上述命令后，下次挂载文件系统时就会启用此设置。若要使此项立即生效，则可以使用 `mount` 命令重新挂载文件系统。

```
$ sudo mount -o remount,noatime /
```

■提醒：有些应用程序需要用到 `atime` 值，因此使用 `noatime` 选项挂载文件系统可能会导致应用程序崩溃。

除了挂载选项，文件系统自身也提供了一些可以用来提升性能的特性。这些特性依据所使用的特定文件系统的不同而不同。其中一个主要的特性是 `dir_index`，它可以应用到 `ext2`、`ext3` 和 `ext4` 文件系统中。启用这些特性后，文件系统会创建内部索引来加速对包含大量文件和子目录的目录的访问。可以使用 `tune2fs` 工具来检查这些特性是否已启用。

```
$sudo tune2fs -l /dev/md0 | grep features
Filesystem features: has_journal resize_inode dir_index filetype needs_recovery ➡
sparse_super large_file
```

可以看出这里已经启用。如果没有，可以使用 `tune2fs` 来启用。

```
sudo tune2fs -O dir_index /dev/md0
```

相反，在特性前加上插入符 (^) 前缀就会将其禁用。

```
sudo tune2fs -O ^dir_index /dev/md0
```

表 17-3 列出了可以提升文件系统性能的大多数有用的选项。

表 17-3 ext2、ext3 和 ext4 必备文件系统特性

特性	适用文件系统	作用
dir_index	ext2, ext3, ext4	加速包含很多文件的目录访问
extents	ext4	预留连续磁盘空间的能力
has_journal	ext3, ext4	日志文件系统，快速出错恢复能力
sparse_super	ext2, ext3, ext4	减少超级备份块的数量，节省空间
uninit_bg	ext4	创建未用节点列表的能力，以便于使用 fsck 命令时可以跳过这些节点

通过确保在适当的文件系统中设置了这些选项，可以最有效地对其进行利用。一旦改变文件系统特性后，`tune2fs` 通常会提示需要运行一次 `e2fsck` 命令，这是因为其中的有些选项的生效需要改变磁盘数据结构。在某个磁盘上运行 `e2fsck` 命令时，要确认没有挂载受影响的分区。

可以通过修改文件 `/etc/mke2fs.conf` 中的默认值来设置各种 `ext` 文件系统中哪些项将被启用。

17.5 小 结

本章中讲述了可以用来很容易判断一个正在运行的主机的健康状况的简单工具。使用这些工具可以做以下事情。

- 查看 CPU 利用率。
- 查看内存和交换空间利用率。

本章还介绍了一些高级系统监控的概念和工具，比如 `SNMP` 和 `Cacti`，使用这些工具可以在持续监测的基础上监控资源的利用率和系统性能。在这里学到了以下 3 点。

- 安装与配置 SNMP。
- 安装与配置 Cacti。
- 使用 Cacti 监控主机的健康度。

下一章将介绍如何对主机和服务的一些监控进行配置。还会介绍如何配置日志记录以及如何对日志中的不正常或错误活动进行监控。

第 18 章 日志记录与监控



James Turnbull 编写

在本书中，已经讨论过日志记录、监控及其在应用程序和服务故障排查方面的重要价值。本章的第一部分中将会详细讨论日志是如何工作的，以及如何对日志记录中所记录的数据加以利用。其中包括了如何对日志条目及日志文件进行存储、汇总、分析、发出报警以及循环记录等内容。还会介绍一些方便与日志文件进行交互的实用工具。

本章的第二部分将会讨论如何使用一个叫 Nagios 的开源工具对前面介绍过的应用程序和服务进行监控。在 Nagios 中，首先可以对主机及主机上运行的服务进行定义，然后还可以确定主机是否已启动以及服务运行状况是否良好。如果运行不正常，监控系统会提示发生了什么问题。此进程将会大大节省管理员辨别和解决问题所花的时间。

18.1 日志记录

读者应该在本书前面的部分中已经看到过，很多应用程序和工具都会对自身的运行和状态数据信息进行记录，这些数据最后都被存到了 `/var/log` 目录下的不同文件中。然而，这些数据并不总是由应用程序直接放到这些文件中的，Linux 还使用了一个叫 `syslog` 的守护进程来承担日志记录的任务。

应用程序把守护进程能够解析的、含有日志条目的特殊格式数据输出到 `syslog` 守护进程中。然后，守护进程收到日志条目并对其进行各种不同的处理，包括将其写入一个文件中。

在本书前面的内容中已经介绍过几个系统日志条目了，下面再来看一下系统日志中的一行，如下所示。

```
Feb 11 08:14:02 au-mel-ubuntu-1 syslogd 1.5.0#2ubuntu6: restart.
```

如上所示，一条 `syslog` 条目由记录日期、记录此条目的主机名称以及记录的数据本身所组成。这里显示的是 `syslog` 守护进程重启的记录 “`syslogd 1.5.0#2ubuntu6: restart`”。

syslog 是普遍使用的 Unix 格式日志，它广泛应用在各种风格的 Linux 和几乎所有风格的 Unix 系统上，也可以使用第三方工具将其应用在 Windows 系统上，而且很多诸如防火墙、路由或交换机之类的网络设备也都能生成 syslog 日志信息。因此 syslog 格式就成了现存的最接近普遍使用的日志标准的格式。

■提示：RFC3164 记录了 syslog 的核心功能，请参考“<http://www.ietf.org/rfc/rfc3164.txt>”。

syslog 格式为各种不同功能不同复杂度的工具所使用，而且这些工具一般都统称为 syslog 守护进程。这些守护进程包括基本的 syslog 工具，也包括比较高级的诸如 syslog-NG（NG 意思是下一代，Next Generation）或 rsyslog 之类的不同工具。

因为基本的 syslog 工具在 Red Hat 和 Ubuntu 中都是默认使用的，所以在这里将介绍基本的 syslog 工具，而且这个基本的工具也为理解日志记录工作在 Linux 系统中的工作原理奠定了基础。

syslog 工具是设计用来产生、处理以及保存有意义的事件警告消息的，这些消息会为管理员管理系统提供必要的信息。syslog 包含一系列程序和库，包括 syslogd、syslog 守护进程以及通信协议。

syslog 中最常用到的部分就是 syslogd 守护进程，此守护进程从系统启动时起就开始运行并监听操作系统和应用程序中发出的消息。要注意到 syslog 守护进程是一个被动的工具，这一点很重要。它只是等待设备或程序向其输入信息，从不主动出击自动地搜集信息。

■注：syslog 还使用另一个守护进程——klogd，也就是内核日志守护进程，此守护进程专门收集从内核中来的消息。它在 Red Hat 和 Ubuntu 中都默认随系统启动。

日志工具的下一个主要部分是 syslog 通信协议。有了这个协议，syslog 就可以通过网络把日志数据发送到一个远程系统中，再由此远程系统的另一个 syslog 守护对日志数据进行进程收集和集中管理。

■提示：syslog 通常使用 UDP 协议并通过 514 端口传送数据。

18.1.1 配置 syslog

syslog 守护进程由一个位于/etc 下名为 syslog.conf 的配置文件进行控制。此文件中包含了 syslog 正在监听什么设备或程序、信息保存于何处以及收到信息后如何处理等内容。

下面是 Ubuntu 中 syslog.conf 文件的默认配置。

```
# /etc/syslog.conf Configuration file for syslogd.
#For more information see syslog.conf(5)
#manpage.
# First some standard logfiles. Log by facility.
auth,authpriv.*      /var/log/auth.log
*.*;auth,authpriv.none -/var/log/syslog
#cron.*               /var/log/cron.log
```



```

daemon.*                -/var/log/daemon.log
kern.*                  -/var/log/kern.log
lpr.*                   -/var/log/lpr.log
mail.*                  -/var/log/mail.log
user.*                  -/var/log/user.log

# Logging for the mail system. Split it up so that
# it is easy to write scripts to parse these files.
#
mail.info               -/var/log/mail.info
mail.warning            -/var/log/mail.warn
mail.err                /var/log/mail.err

# Logging for INN news system
#
news.crit               /var/log/news/news.crit
news.err                /var/log/news/news.err
news.notice             -/var/log/news/news.notice

# Some 'catch-all' logfiles.
#
*.=debug;\
auth,authpriv.none;\
news.none;mail.none    -/var/log/debug
*.=info;*.=notice;*.=warning;\
auth,authpriv.none;\
cron,daemon.none;\
mail,news.none          -/var/log/messages

# Emergencies are sent to everybody logged in.
#
*.emerg                 *

# I like to have messages displayed on the console, but only on a
# virtual console I usually leave idle.
#
#daemon,mail.*;\
#news.=crit;news.=err;news.=notice;\
#*.=debug;*.=info;\
#*.=notice;*.=warning   /dev/tty8

# The named pipe /dev/xconsole is for the 'xconsole' utility. To
# use it, you must invoke 'xconsole' with the '-file' option:
#
#   $ xconsole -file /dev/xconsole [...]
#
# NOTE: adjust the list below, or you'll go crazy if you have a
# reasonably busy site..
#
daemon.*;mail.*;\
news.err;\
*.=debug;*.=info;\
*.=notice;*.=warning    |/dev/xconsole

```

我们已经知道，Red Hat 和 Ubuntu 都把日志文件保存在 `/var/log` 目录下并使用不同的文件名来保存不同类型的日志条目，例如，就像在第 10 章看到过的一样（也可以从上述的 `syslog.conf` 配置文件中看出来），Ubuntu 把邮件相关的 `syslog` 条目保存在 `mail.log` 文件中。但在 Red Hat 中，邮件相关的 `syslog` 条目却是保存在了 `maillog` 文件中。可以查看主机的 `syslog.conf` 配置文件以确定所需信息的保存位置。

syslog.conf 文件的每一行都有两个字段构成，一个选择器字段和一个动作字段，两个字段使用空格或制表符分开。列表 18-1 中显示了一个示例行。

列表 18-1 syslog.conf 语法

```
mail.info                                -/var/log/mail.info
```

此例中选择字段为 mail.info，和它一起的处理字段为/var/log/mail.info。选择器字段指定了设施和优先权，用句点隔开。设施指明日志消息的来源，比如 mail 设施用来指定类似 Postfix 的邮件服务相关的日志信息。有很多可用的设施，下一节将会对其进行一一介绍。每一个应用程序都会为自己的日志条目指定所用的设施。

优先权（也就是此例中的 info）通知 syslog 正在传送的消息的重要性。稍后会专门讨论可用的优先权范围。同样地，每次将消息传向 syslog 时，应用程序都会为其指定一个优先权。

动作字段告诉 syslog 如何处理收到的消息，通常的处理方法是将消息写入一个文件。在列表 18-1 中，所有从 mail 设施中收到的带有 info 优先权的消息将会被写入文件/var/log/mail.log 中。

1. 设施

设施用来识别 syslog 信息的来源。有些操作系统的守护进程以及其他一般应用程序的守护进程都有与之相关联的标准设施。mail 和 kern 设施就是两个很好的例子，它们分别对应着邮件相关的事件通知消息和内核相关的消息。

其他没有指定设施的进程和守护进程可以使用 local 设施，此设施有从 local0 到 local7 一共 8 个。表 18-1 列出了所有的 syslog 设施。

■提示：在 Red Hat 系统里，local7 通常为启动消息专用，并将其导入文件/var/log/boot.log 中。

表 18-1 Linux 中的 syslog 设施	
设施	用途
auth	安全相关的消息
auth-priv	访问控制消息
cron	cron 相关的消息
daemon	进程与守护进程消息
kern	内核消息
local0–local7	本地定义消息预留
lpr	后台（打印）系统消息
mail	邮件相关消息
mark	时间戳消息，由 syslogd 产生
news	网络新闻相关消息（如 Usenet）
syslog	syslog 相关消息
user	未指定设施时的默认设施
uucp	UUCP 相关消息

■注：mark 设施是个特例。它由在 `syslogd` 命令后加上 `-m` 标识后产生的时间戳消息所使用。在“启动与配置 `syslog` 守护进程”一节中可以看到于此有关的更多内容。

除此之外，还有两个特别的设施：一个是 `*`，代表所有设施的通配符；另一个是 `none`，否定对一个设施的选择。

列表 18-2 中显示了通配符选择器。

列表 18-2 `syslog.conf` 中的通配符选择器

`*.emerg` `/dev/console`

此设置将会把不管使用何种设施的具有 `emerg` 优先权的所有消息发送到控制台（这是除了将消息写入文件，`syslog` 可能会执行的另外一种动作）。

也可以使用 `none` 通配符选择器来指定不从某个特定的设施中选择消息。列表 18-3 所示的例子中，通知 `syslog` 不再将任何内核消息存入文件 `/var/log/messages` 中。

列表 18-3 `syslog.conf` 文件中的 `none` 通配符选择器。

`kern.none` `/var/log/messages`

2. 优先权

依据消息的重要性进行优先权划分。由低到高排列，分别是 `debug`、`info`、`notice`、`warning`、`err`、`crit`、`alert` 和 `emerg`。每一个优先权选择符应用于其自身所指代的以及所有更高级别的优先权，因此，`uucp.err` 包括所有 `uucp` 设施中具有 `err`、`crit`、`alert` 以及 `emerg` 优先权等级的消息。

和设施一样，优先权中也可以使用通配符选择器 `*` 和 `none`。另外，还可以使用 `=` 和 `!` 修饰符。修饰符 `=` 表示只选中一个优先权，比如，“`cron.=crit`”表示只选择 `cron` 设施中具有 `crit` 优先权等级的消息。修饰符 `!` 作用则是否定的，比如，“`cron.!crit`”表示选择 `cron` 设施中除了具有 `crit` 及更高优先权等级的所有消息。当然也可以把这两个修饰符结合起来产生与 `=` 相反的效果，如“`cron.!=crit`”选择 `cron` 设施中除了具有 `crit` 优先权等级的所有消息。每一个选择器中只能列出一个优先权等级或一个优先权等级通配符。

3. 动作

Actions（动作）字段告诉 `syslogd` 如何处理收到的事件通知消息。`syslog` 可以执行 5 个潜在的动作，如下所示。

- 记录到一个文件。
- 记录到一个设备。
- 记录到一个命名管道。
- 记录到一个特定用户或控制台。
- 将日志发送到其他主机。

列表 18-4 中列出了前 4 个 `syslogd` 可以执行的动作，包括记录到文件、到设备文件、到命名管道以及到控制台或用户屏幕。

列表 18-4 文件、设备、命名管道动作

cron.err	/var/log/cron
auth.!=emerg	/dev/lpr1
news.=notice	/tmp/pipe
auth-priv	root,bob

第一行表示所有 cron 中具有 err 或更高优先权等级的消息都被记录到文件/var/log/cron 中。

■注：记录到文件中时，syslogd 允许在文件名前加上一个破折号 (-)，就像-/var/log/auth，这就告诉 syslog 写入后不要对文件进行同步。此功能是设计用来加速日志写入进程的，但是这也意味着如果在写入的时候系统崩溃，将会发生数据丢失。

第二行将 auth 设施中除了 emerg 优先权等级的所有消息都发送到一个本地打印机 lpr1。

第三行将 news 设施中所有具有 notice 及更高优先权等级的消息都发送到/tmp/pipe 命名管道中。

■注：将消息送入命名管道可以将 syslog 数据发送到其他的应用程序，例如可以使用命名管道收集日志消息，然后将收集到的消息都传给一个日志相关引擎或一个数据库。

第四行，也就是最后一行，如果用户已登录，则将 auth-priv 设施中所有消息发送给用户 root 和 bob。

还有最后一个可以执行的动作，那就是将日志发送到另一个主机，如列表 18-5 所示。

列表 18-5 记录到一个远程系统

mail	@headoffice.example.com
------	-------------------------

此例中，所有 mail 设施中的消息都被发送到了 headoffice.example.com 主机中。

若要传送所有日志，可以使用以下语法。

```
*.* @headoffice.example.com
```

syslog 使用 UDP 协议 514 端口传送日志信息。这里假设已经将远程主机上的 syslog 守护进程配置为可以接收日志，同时假设在接收日志条目的位置配置了合适的防火墙规则。比如：

```
-A Firewall-eth0-INPUT -p udp -m state --state NEW -s 192.168.0.254--dport 514 ➡  
-j ACCEPT
```

这样就创建了一个防火墙规则，此规则允许主机接收从 IP 地址为 192.168.0.254 的主机 UDP514 端口上发来的 syslog 数据。

4. 合成多项选择器

还可以在 syslog.conf 文件中合成多项选择器，以允许进行复杂的选择和过滤。比如，可以在一个选择器中列出用逗号隔开的多个设施（如列表 18-6 所示）。

列表 18-6 多项设施

auth,auth-priv.crit	/var/log/auth
---------------------	---------------

上述设置将所有的 auth 消息和所有具有 crit 及更高优先权等级的 auth-priv 消息写入/var/

log/auth 文件中。

像这样的设置是不能使用优先权等级来实现的。如果要列出多个优先权等级，可以列出使用分号隔开的多项选择器，如列表 18-7 所示。

列表 18-7 多个优先权

```
auth;auth-priv.debug;auth-priv.!=emerg      /var/log/auth
```

此例显示了如何将除了具有 emerg 优先权级别的 auth-priv 消息的所有 auth 消息和所有具有 debug 及更高级别优先权的 auth-priv 消息写入/var/log/auth 文件中。

提示：只要记住，多项选择器的筛选顺序是从左到右的，syslog 将从最左边的选择器开始处理，然后顺序处理右边的每一个选择器。考虑到这一点后，只要将最宽泛的筛选条件放置在最左边，然后向右逐次缩小筛选条件就可以了。

也可以设置多行来将消息发送到不同的位置，如列表 18-8 所示。

列表 18-8 将日志记录到多个位置

```
auth      /var/log/auth
auth.crit  bob
auth.emerg /dev/console
```

这里和前面一样把 auth 消息记录到了文件/var/log/auth 中，同时，如果用户 bob 已经登录了，也把具有 crit 及更高优先权级别的 auth 消息发送给它，同时还要把其中具有 emerg 优先权等级的消息发送到控制台。

18.1.2 启动与配置 syslog 守护进程

syslogd 守护进程及其姊妹进程（klogd 守护进程）通常在系统启动时就启动了。在 Red Hat 中，此过程由启动 syslogd 和 klogd 的脚本 syslog init 完成。在 Ubuntu 中，syslog 守护进程由脚本 syslogd init 启动。

在 Red Hat 和 Ubuntu 中都可以分别使用文件/etc/sysconfig/syslog 或/etc/default/syslogd 对 syslogd 的一些选项进行自定义。下面介绍其中的几个选项。

前面讨论过 mark 设施的消息。日志中有很多如下所示的在指定的时间间隔生成的时间戳记录。

```
Feb 24 21:46:05 au-mel-ubuntu-1 -- MARK -
```

除了其他原因，这些消息仅作为标识对解析日志文件的程序就是很有用的。当启动 syslogd 时添加“-m mins（秒数）”选项就可以产生这些时间戳。若想每 10 分钟产生一个 mark 消息，就要在/etc/sysconfig/syslog 或/etc/default/syslogd 文件中加入“-m 10”选项。这里，已经在/etc/default/syslogd 文件中添加了此选项。

```
# Top configuration file for syslogd
#
# Full documentation of possible arguments are found in the man page
# syslogd(8).
#
```



```
# For remote UDP logging use SYSLOGD="-r"
#
SYSLOGD="-m 10"
```

请记住，**mark** 本身也是一个设施，可以将其输出定向到一个特定的文件或目的地中。

```
mark    /var/log/messages
```

这样，所有的 **mark** 设施消息都将被定向到 `/var/log/messages` 文件中。Red Hat 和 Ubuntu 中默认的 **syslogd** 设置是使 “-m” 选项等于 0，从而禁止 **mark** 消息的产生。

最后，看一下 “-r” 选项，此选项允许 **syslogd** 在 UDP 端口 514 上接受外部源的消息。如果 **syslogd** 带有 “-r” 选项启动，它将监听从 UDP 端口 514 上来的 **syslog** 消息，然后将收到的消息写入到主机上合适的日志文件中。

在 Red Hat 主机上的 `/etc/sysconfig/syslog` 文件中可以看到此项已被启用。

```
# Options to syslogd
# -m 0 disables 'MARK' messages.
# -r enables logging from remote machines
# -x disables DNS lookups on messages recieved with -r
# See syslogd(8) for more details
SYSLOGD_OPTIONS="-m 0 -r"
# Options to klogd
# -2 prints all kernel oops messages twice; once for klogd to decode, and
#   once for processing with 'ksymoops'
# -x disables all klogd processing of oops messages entirely
# See klogd(8) for more details
KLOGD_OPTIONS="-x"
#
SYSLOG_UMASK=077
# set this to a umask value to use for all log files as in umask(1).
# By default, all permissions are removed for "group" and "other".
```

很多 **syslogd** 守护进程启动时都默认不启用 “-r” 项，因此如果想让 **syslogd** 进行监听需要专门启用此项。

将 **syslogd** 对网络开放可能是一件危险的事情。**syslogd** 守护进程对从何处接收消息并不具有选择性，没有访问控制，网络中的任何系统都可以向 **syslogd** 端口写入日志信息。这将把主机暴露于 DoS 攻击的危险之下，或者一个恶意程序可以使用消息将系统淹没并将占用日志分区上的所有空间。

如果要使用 **syslogd** 进行远程日志记录，有几个方法可以使安装更加安全。对 **syslogd** 守护进程最明显的威胁就是 DoS 攻击，受到此攻击时系统被可能会填满磁盘上所有空间的消息所淹没。如果日志文件位于 **root** 分区上，则系统很有可能会崩溃，为了减少这种潜在崩溃的可能性，建议将日志文件保存在一个非 **root** 分区（非/分区）上，这就意味着即使磁盘上的所有空间都被消耗掉了，系统也不会崩溃。

第二个增强用于远程日志记录的 **syslogd** 安全性的方法是确保防火墙规则只允许从那些要传送日志数据的系统来的连接请求。千万不要将 **syslogd** 守护进程对所有进入连接开放。使用一个适当的防火墙规则，例如前面创建的一个。

```
-A Firewall-eth0-INPUT -p udp -m state --state NEW -s 192.168.0.254--dport 514 ➡
-j ACCEPT
```


18.1.3 使用 logger 工具测试日志记录

Red Hat 和 Ubuntu 中自带的 logger 是测试日志记录配置的一个很有用的工具。

```
$ logger -p mail.info "This is a test message for facility mail and priority info"
```

此命令将会把消息 “This is a test message for facility mail and priority info” 传送到 syslog 守护进程，然后写入为具有 info 优先权等级的 mail 设施消息所配置的日志文件中。

可以看到，“-p” 参数可以指定一个设施和优先权等级的组合，随后的测试消息要放在双引号中。

出于测试的目的，为了能产生多条消息，通常把 logger 工具放在 bash 脚本中使用。列表 19-9 中的脚本将为每一个设施和优先权等级产生一个 syslog 消息。

列表 18-9 日志测试 bash 脚本

```
#!/bin/bash
for f in
{auth,authpriv,cron,daemon,kern,lpr,mail,mark,news,syslog,user,uucp,local0,local1,
local2,local3,local4,local5,local6,local7}
do
  for p in {debug,info,notice,warning,err,crit,alert,emerg}
  do
    logger -p $f.$p "Test syslog messages from facility $f with priority $p"
  done
done
exit 0
```

还可以使用 logger 工具把一个持续增长的文件导入 syslog 中。

```
$ tail -f /tmp/logfile | logger -p daemon.info
```

这里将跟踪到的/tmp/logfile 文件的内容引入 logger 命令中。这样，文件中的每一行都将被写入具有 info 优先权等级的 daemon 设施中。

18.1.4 日志管理与轮替

管理日志环境的一个重要内容是控制日志文件的体积并将其保持在可管理的大小。要做到这一点，可以轮替日志内容。

日志轮替是周期性地复制日志文件的处理过程，通常复制时要加上日期或递增的数字前缀，然后 syslog 守护进程将把日志记录到一个新的文件中。通常应该按照固定的周期轮替日志文件，如一周或一月一次。

看一个示例。就拿/var/log/mail.log 文件来说，可以每天轮替一次此文件并将轮替下来的文件保存 7 天。日志轮替进程将会在设定的时间自动启动，将现存的 mail.log 文件复制为 mail.log.1 文件，然后再创建一个空的 mail.log 文件。日志轮替进程将会自动增加轮替文件后缀的数字，若文件 mail.log.1 存在了，那它会被重命名为 mail.log.2，依此类推。如果轮替时有一个名为 mail.log.7 的文件，则此文件将被删除，而 mail.log.6 文件将被提升为 mail.log.7。

对手动管理来讲，日志轮替可能会是相当复杂的，所以建议使用 `logrotate` 工具来进行。`Red Hat` 和 `Ubuntu` 都自带 `logrotate` 工具，而且通常已经安装并配置过了。默认的配置就可以处理主机上安装的大多数应用程序的典型日志文件。

`logrotate` 命令的配置很简单，它需要依靠定时任务工具（`crontab`）才能运行在计划模式下。基本的 `logrotate` 配置文件位于 `/etc/logrotate.conf`，列表 18-10 列出的是一个典型的配置文件的内容。

列表 18-10 `logrotate.conf`

```
#log rotation
weekly
# keep old logs
rotate 4
#create new logs
create
#include .d files
include /etc/logrotate.d
```

此示例文件包含了 `logrotate` 工具用来处理日志文件的全局选项。在这个示例文件中规定，所有的日志文件每周轮替一次，日志文件删除前最多轮替 4 次，之后自动创建新的日志文件，而且 `logrotate` 工具会检查 `logrotate.d` 目录下的任何新建 `logrotate` 文件。还可以使用其他选项，表 18-2 中列出了一些可用的选项，其他选项的相关内容则可以深入研究 `logrotate` 的 `man` 文件。

表 18-2 `logrotate.conf` 选项

选项	说明
<code>daily</code>	日志每天轮换一次
<code>weekly</code>	日志每周轮换一次
<code>monthly</code>	日志每月轮换一次
<code>compress</code>	旧的日志文件使用 <code>gzip</code> 进行压缩
<code>create mode</code>	使用本地 <code>0700</code> 模式、用户模式、用户组模式或组模式创建新的日志文件（除此之外就是不创建）
<code>ifempty</code>	即使日志文件为空也轮换
<code>include directory or filename</code>	包括列出的文件和被 <code>logrotate</code> 压缩的目录
<code>mail address</code>	当日志文件轮换出时，将被发送到此处指定的邮箱中
<code>nomail</code>	轮换出的日志文件不发送邮件
<code>missingok</code>	若日志文件丢失， <code>logrotate</code> 将跳过此文件并指向下一个文件，不发出任何出错信息
<code>nomissingok</code>	日志文件丢失时，产生出错信息（默认选项）
<code>rotate count</code>	设置日志文件删除前轮换的次数。若次数设置为 0，则旧的日志文件不经轮换，直接删除
<code>size size[M,k]</code>	当日志文件大小大于此处设置时开始轮换。 <code>M</code> 指兆字节， <code>k</code> 指千字节
<code>sharedscripts</code>	轮换的日志都可以在轮换前或轮换后执行指定的脚本。若日志文件定义中包含了一个日志文件集合（如 <code>/var/log/samba/*</code> ），同时设置了 <code>sharedscripts</code> ，则执行一次轮换前或轮换后脚本，否则设置为 <code>nosharedscripts</code>

列表 18-10 中的最后一个命令主要是用来驱动 `logrotate` 的 `includes` 命令。

在列表 18-10 中引入的 `logrotate.d` 目录中有一个文件集合，用来告诉 `logrotate` 命令如何

处理各种不同日志文件。

虽然也可以定义另外的路径和文件并将其引入 `logrotate.conf` 文件以适应各种日志环境，但很多发行版都默认使用 `logrotate.d` 路径及路径下一定数量的预定义文件来处理如 `mail`、`cron` 和 `syslog` 消息等常见的日志轮替。建议无论添加什么新的日志轮替文件，都要在这儿添加。

■注：很多软件包在安装时也会把日志轮替文件放在此目录中。

列表 18-11 列出了其中一个文件的内容。

列表 18-11 Red Hat syslog 日志轮替文件

```
/var/log/messages /var/log/secure /var/log/maillog /var/log/spooler ➡
/var/log/boot.log /var/log/cron
{
daily
rotate 7
sharedscripts
postrotate
    /bin/kill -HUP 'cat /var/run/syslogd.pid 2> /dev/null' 2> /dev/null || true
endscript
}
```

可以在这些文件中重新定义 `logrotate.conf` 文件中的很多全局选项的值来对特殊的文件或目录自定义日志轮替规则。列表 18-11 首先列出了要进行轮替的所有文件，可以在这里使用语法 `/path/to/log/files/*` 来包含多个目录。

然后是封闭在 `{}` 中的对指定文件集合进行处理的诸多选项。此例中，将全局日志选项重新定义为对指定的文件每天轮替一次并保留 7 个日志轮替文件。

下面来运行此脚本。可以使用 `prerotate` 命令，此命令在轮替任何日志文件之前运行脚本；或者使用 `postrotate` 命令，此命令在轮替日志文件之后运行脚本。

列表 18-11 显示的是一个日志文件轮替完成后重启 `syslog` 守护进程的脚本。由于启用了 `sharedscript` 选项，则无论有多少单个的日志文件被轮替，此脚本只运行一次。`endscript` 选项结束脚本的运行。

那么 `logrotate` 是如何运行的呢？在 Red Hat 和 Ubuntu 中都默认使用 `cron` 命令按计划运行 `logrotate` 命令。当然，也可以在命令行中手动运行。

如果在命令行中运行，`logrotate` 命令默认调用 `/etc/logrotate.conf` 配置文件，也可以在运行时使用如下格式改变默认的配置文件的。

```
$ sudo logrotate /etc/logrotate2.conf
```

`logrotate` 命令还有几个命令行运行选项，如表 18-3 所示。

表 18-3 logrotate 命令行选项

选项	说明
-d	调试模式，不对日志文件作任何改变；输出轮替操作的内容。也可以作为详细模式
-v	详细模式
-f	即使没有必要也强制轮替日志文件

很多系统上，logrotate 都默认由 cron 设置为每天运行一次的模式，建议读者也要使用这种模式。

18.2 日志分析与关联

现在有了很多的日志文件了，那究竟可以使用它们做些什么呢？日志主要有两个用途：

- 识别问题的出现；
- 帮助分析所出现的问题。

实现第一个目的需要一个工具，这个工具将会对特殊的日志消息进行识别并在问题出现时发出示警。此进程称为日志分析与关联，大家经常认为此进程具有魔法。下面介绍一个叫 SEC 的工具，此工具会使日志分析与关联成为日常监控事务中非常简单的一部分。

首先应该明白，分析和关联是截然不同的两件事情。分析是指研究各个组成部分及其构成整体的内部联系。作为一个系统管理员，最好的分析工具就是管理员自己。系统管理员通晓主机运行模式并常常能比自动监控或示警系统更快地对同一问题做出反应。

但是人工检测的模式有两个问题。第一个问题是管理员不可能同时对所有可能出现问题的位置进行监控，第二个问题是系统数据收集的大小可能出现不可控制的局面。

鉴于此，关联出现了。关联是一种精心定义的用于探测数据之间相互关系的行为。首先设置收集数据的工具，过滤信息如“谷壳中的小麦”，然后将余下的信息进行关联，把恰当的信息显示在管理员的面前，这样就能进行更为准确的分析了。

经过合理设置和管理的工具可以对主机日常运作产生的不断增长的数据进行排序。这些工具可以检测数据之间的相互关系，然后要么把这些数据块组合成一个连贯的整体，要么把一些适当的数据块交给管理员，然后让管理员自己对其进行组合分析。

但使用工具之前必须确保工具的适用性并对其进行配置以寻找恰当的信息，这样才能依赖于它们来汇报一些需要管理员进行干预的出错事件。

构建这样一个自动日志监控系统的第一步是确定收集的是恰当的信息并将其放置于恰当的位置。把所有应用程序、设备及主机和它们存入日志的位置做个列表。第二步是把所有的信息放在一起并找出确实需要知道的，再把关键信息及对主机很重要的信息做个列表。

把这些列表根据重要性进行分组，有的信息可能希望以短消息的方式进行通知，有的可以通过 e-mail 的方式通知，而有的可能要激活一个自动处理的进程或触发尝试自我恢复，如进程重启等。

第三步是部署日志关联与分析，包括配置关联工具和设计所需要的反应。一定要确保记录了每一条消息、对此消息的反应以及任何与此消息有关的特殊信息。

18.2.1 SEC 简介

SEC，或叫简单时间关联器，是现有的最强大的开源日志关联工具。SEC 利用 Perl 正则表达式从多数 Linux 系统都会产生的大量日志记录中寻找对系统运行有重要意义的消息。它

可能会找到单独的一条消息，也可能会匹配到一组相关的消息，比如，它可以找到与表明用户登入或登出系统相匹配的消息。SEC 也可以对收到的消息进行计数并只有在消息数量超过设定的限制值时才采取行动。SEC 还可以通过执行一些操作对收到的消息作出反应，比如运行一个 shell 脚本程序等，而这些操作中可能会包含消息的部分内容。例如，可能会运行一个 shell 脚本程序作为 SEC 操作，而且会使用一部分或全部消息内容作为变量放到 shell 脚本程序中。

正则表达式

由于 SEC 依赖于 Perl 的正则表达式，因此读者需要习惯正则表达式的使用。乍看上去，这些表达式可能会显得比较可怕，但是学习和使用它们却真的很简单。正则表达式是一个简单的技术，用来从大量的数据集合中识别出用户感兴趣的文本或数字。一个简单的应用可能是无论出现 cat 这个词都要对其进行匹配。使用正则表达式，可以在一个词出现时对其进行全字匹配或作为其他词的一部分进行匹配（比如 category）。

有很多各种各样的正则表达式，同时不同的编程语言所使用的语法都略微不同。Perl 编程语言的正则表达式语法应用很广泛而且很容易学习，可以从参考 Perl 手册的正则表达式部分开始，里面的描述的确很棒。可以参考“<http://www.perldoc.com/perl5.6.1/pod/perlre.htm>”和“<http://www.perldoc.com/perl5.8.0/pod/perlretut.html>”，还可以参考很多非常精彩的正则表达式参考书，例如 Jeffery Friedl 编写的《精通正则表达式》（O'Reilly, 2006），还可以在以下网站上找到其他相关资料。

- <http://www.regextester.com/>：允许测试正则表达式的在线测试工具。
- <http://www.troubleshooters.com/codecorn/littperl/perlreg.htm>：教程。
- <http://www.cs.tut.fi/~jkorpela/perl/regexp.html>：教程。
- <http://www.quanetic.com/Regex>：正则表达式测试。
- <http://www.perlfect.com/articles/regextutor.shtml>：正则表达式互动教程。

看到所有这些功能，大家可能会觉得这简直有点杀鸡用牛刀了，但是扩展事件关联功能的能力远远超过了实施的成本。根据经验，在构建日志记录环境时关键的一点是不能做出任何可能会导致重要信息丢失的妥协，不过 SEC 的丰富功能应该能够涵盖目前及将来的事件关联需求了。

出于 SEC 的复杂性，在本章中对其所有特性进行全面的讨论是不可能的，因此，本章将不会涉及 SEC 某些特定情况下的比较高级的特性。SEC 的全面部署及其中的变量的相关内容都可以很轻松地单独成书。本章将从以下诸方面开始介绍 SEC，即如何安装 SEC，如何运行 SEC，如何使日志指向 SEC，以及如何设置一些基本的消息匹配规则，然后列出一些可以帮助读者在自己的环境中完全启用 SEC 的资料。

提示：开始学习 SEC 更多内容的好去处是 SourceForge 站点上的 SEC 邮件列表。可以订阅邮件列表并到“<http://lists.sourceforge.net/lists/listinfo/simple-evcorr-users>”链接处阅读档案内容。SEC 的作者（Risto Vaarandi）是此邮件列表专任的、活跃的而且是非常乐于提供帮助的参与者，在邮件列表档案中包含了很多有用的 SEC 规则，对读者学习 SEC 将会非常有帮助。

18.2.2 安装 SEC

可以使用软件包或通过 SEC 网站安装 SEC。在 Ubuntu 上，需要一个名为 sec 的软件包。

```
$ sudo aptitude install sec
```

在 Red Hat 上，SEC 应用程序已经被 EPEL 项目（此项目为 Red Hat Enterprise Linux 提供附加的软件包）打包，可以下载并使用 RPM 安装。

```
$ sudo rpm -Uvh http://download.fedora.redhat.com/pub/epel/5/i386/
sec-2.4.1-1.el5.noarch.rpm
```

■注：也可以在主机中添加 EPEL 库并通过 yum 使用下述链接处提供的命令进行安装，见“http://fedoraproject.org/wiki/EPEL/FAQ#How_can_I_install_the_packages_from_the_EPEL_software_repository.3F”。

还可以从 SEC 的主页下载部分下载 SEC，例如：

```
$ wget http://prdownloads.sourceforge.net/simple-evcorr/sec-2.5.0.tar.gz
```

下载后安装 SEC 的过程非常简单。首先解压档案文件并进入解压后的目录。

```
$ tar -zxf sec-2.5.0.tar.gz
$ cd sec-2.5.0
```

在档案文件内部是 SEC 工具的引擎，一个名为 sec.pl 的 Perl 脚本文件。将 sec.pl 脚本文件拷贝到用户路径下的一个目录中，改变其模式为可运行，所有者为 root 用户。

```
$ sudo cp sel.pl /usr/local/bin/sec.pl
$ sudo chmod 0755 /usr/local/bin/sec.pl
$ sudo chown root:root /usr/local/bin/sec.pl
```

SEC 还自带了一个很好理解的 man 页面，也运行如下命令进行安装。

```
$ sudo cp sec.pl.man /usr/local/share/man/man8
$ sudo chown root:root /usr/local/share/man/man8/sec.pl.man
```

18.2.3 运行 SEC

可以通过在命令行中运行 sec 二进制文件启动 SEC（如果是源代码方式安装应该运行 sec.pl 脚本文件），或者使用 init 脚本启动它。

在 Ubuntu 和 Red Hat 中，由软件包安装的 init 脚本文件名都是 sec，而两个文件都需要进行稍微的配置才能启动 SEC。

1. 在 Red Hat 上运行 SEC

简单看一下在 Red Hat 中的/etc/sysconfig/sec 文件。

```
# Because SEC usage varies so widely from user to user, it is configured by
# default to not run. Please read 'sec --help' for valid options to use in
# this configuration directive, or use the sample defaults included below.
```



```
#
# If you would like to run multiple instances of sec in order to track more
# than one log file, you can use also use $SEC_OPTIONS as an array.
#
# Also, please don't forget to read the sec man page or look at the
# configuration options for /etc/sec/.
#
# Default:
#
# SEC_ARGS="-detach -conf=/etc/sec/*.sec -input=/var/log/messages ➡
-log=/var/log/sec -intevents -pid=/var/run/sec.pid"
#
# For Multiple instances of SEC, use something like:
#
# SEC_ARGS[0]="-detach -conf=/etc/sec/sys/*.sec -input=/var/log/messages ➡
-log=/var/log/sec -intevents -pid=/var/run/sec.sys.pid"
#
# SEC_ARGS[1]="-detach -conf=/etc/sec/mail/*.sec -input=/var/log/messages ➡
-log=/var/log/sec -intevents -pid=/var/run/sec.mail.pid"
```

在这个文件中，需要告诉 SEC 使用什么启动选项。最简单的方式就是使用已经在文件中指定的 SEC_ARGS 行，要使用此行，直接取消注释就可以了。

```
SEC_ARGS="-detach -conf=/etc/sec/*.sec -input=/var/log/messages -log=/var/log/sec ➡
-intevents -pid=/var/run/sec.pid"
```

仔细研究一下此行，里面的每一个选项都会被传递给 SEC。-detach 选项将使 SEC 作为一个守护进程运行。-conf 选项加载 SEC 配置文件，此例中也就是保存用来匹配特定日志条目的规则的文件。在 Red Hat 中，SEC 会加载位于 /etc/sec 目录下的所有带 .sec 后缀的文件。

-input 选项告诉 SEC 监控那个文件中的 syslog 日志条目。示例行中，只指定对文件 /var/log/messages 进行查找，也可以指定一个文件通配表达式或使用多个 -input 选项来指定多个文件。例如，要指定所有带 .log 后缀的文件，可以如下所示进行设置。

```
-input *.log
```

或可以这样指定多个文件。

```
-input=/var/log/messages -input=/var/log/secure
```

-log 选项告诉 SEC 将其输出存到何处，-intevents 选项产生 SEC 应用程序自身的启动、停止以及重启日志事件。

最后，-pid 选项指定 sec 守护进程的 PID 写到哪里。

可以通过以下设置使 SEC 随系统启动。

```
$ sudo chkconfig --level 35 sec on
$ sudo chkconfig --add sec
```

注：目前还没有定义任何规则，因此 SEC 启动时将不进行任何操作。

2. 在 Ubuntu 上运行 SEC

在 Ubuntu 中，SEC 的 8.04LTS 版本缺少了部分内容，例如 init 脚本等。为下一个 Ubuntu

版本准备的 `sec` 软件包就包含了所有所需的功能，因此要从最新的发行版本中提取 `/etc/default/sec` 文件和 `/etc/init.d/sec` 文件。要提取这些文件，总需要到最新的发行版本中下载以下部分内容。

```
$ cd /tmp
$ wget http://archive.ubuntu.com/ubuntu/pool/universe/s/sec/sec_2.4.2-1.diff.gz
$ gunzip sec_2.4.2-1.diff.gz
$ patch -p1 < 2.4.2-1.diff
$ cp debian/sec.default /etc/default/sec
$ cp debian/sec.init /etc/init.d/sec
```

首先，切换到 `/tmp` 目录下。然后从构成 `sec` 软件的包的文件中下载一个文件并用 `gunzip` 命令将其解压缩。`patch` 命令是一个可以进行查找并替换文件操作的特殊命令。在这里，此命令在一个名为 `debian` 的目录下创建了几个 Ubuntu 安装软件包使用的文件，包括即将用到的 `sec.default` 文件和 `sec.init` 文件。随后，把这两个文件复制到适当的位置并重命名。

■注：本书的源代码中已经包含了上述文件。

现在就有了可以用来矫正 `/etc/default/sec` 中配置的文件了。

```
#Defaults for sec
RUN_DAEMON="no"
DAEMON_ARGS="-conf=/etc/sec.conf -input=/var/log/syslog -pid=/var/run/sec.pid ➡
-detach -syslog=daemon"
```

首先，把 `RUN_DAEMON` 选项设为 `yes` 以启用 `sec` 守护进程。

```
RUN_DAEMON="yes"
```

接着，需要升级一下要传递给 `sec` 守护进程的参数。把 `DAEMON_ARGS` 行升级为如下所示。

```
DAEMON_ARGS="-detach -conf=/etc/sec/*.sec -input=/var/log/syslog ➡
-input=/var/log/daemon.log -input=/var/log/messages -input=/var/log/auth.log ➡
-input=/var/log/user.log -pid=/var/run/sec.pid -syslog=daemon"
```

第一个选项 “`-detach`” 使 `SEC` 运行在守护进程的模式下。下一个选项 “`-conf`” 装载位于 `/etc/sec` 目录下所有带 `.sec` 后缀的文件中的 `SEC` 运行配置。现在就快速创建此目录，如下所示。

```
$ sudo mkdir /etc/sec
```

目前还没有任何规则，因此 `SEC` 启动时不进行任何操作。

`-input` 选项告诉 `SEC` 为 `syslog` 消息监听那些文件，这里指定了严格用来输出日志记录数据的主要文件，比如 `/var/log/syslog` 和 `/var/log/daemon.log`。

最后，`-pid` 和 `-log` 选项告诉 `SEC` 其自身的 `PID` 文件为 `/var/run/sec.pid`，同时将其自身的日志条目写入文件 `/var/log/sec.log` 中。

现在可以设置 `SEC` 随 Ubuntu 系统启动了。

```
$ sudo update-rc.d sec defaults
```

3. SEC 命令行选项

`SEC` 提供了一些附加的命令行选项来控制其自身的运行和配置。表 18-4 中列出了其中一

些比较重要的选项。

表 18-4 SEC 命令行选项	
选项	描述
-input = file pattern[= context]	此处指定 SEC 的输入源，可以是文件、命名管道或标准输入设备。可以在命令行中指定多个 input 项。可选的 context 项用于设置一个上下文环境，上下文环境可以帮助用户写出匹配特定输入源事件的规则。本章没有涉及上下文环境的相关内容
-pid = pidfile	此选项用来指定存储 SEC 进程 ID 的文件。如果需要一个 PID 文件，必须使用此选项
-quoting and -noquoting	若启用了 quoting 选项，SEC 发送到外部 shell 命令中的字符都将使用双引号包括起来。默认不启用此项
-tail and -notail	此选项告诉 SEC 如何处理文件。若设置了 -notail 选项，SEC 将会读取任何输入源并在文件或输入源结束时退出。若设置了 -tail 选项，SEC 将跳到输入源的结尾并等待新的输入，这与执行了命令“tail -f”一样。默认是启用 -tail 项
-fromstart and -nofromstart	此项与 -tail 结合使用。当使用 -fromstart 时，强制 SEC 从头至尾处理输入文件，然后进入 tail 模式等待新的输入。显然，此项对 -notail 选项无作用。默认选项为 -nofromstart
-detach and -nodetach	若在命令行中添加了 -detach 项，SEC 将作为一个守护进程运行。默认为 -nodetach，此时 SEC 运行作为一个可控终端
-testonly and -notestonly	若指定了 -testonly 选项，检测到配置文件中的任何错误时，SEC 都将立即退出运行。若配置文件中没有错误，SEC 将在退出时返回代码 0；否则退出时返回代码 1。默认设置为 -notestonly

可以在 SEC man 页面输入资源部分的 time-outs 中找到有关附加的 SEC 选项的内容。

18.2.4 使用 SEC

下面演示一下如何创建一些简单的 SEC 规则，以便于读者理解 SEC 是如何运行的。SEC 的每一个配置文件都包含了一系列规则声明，每一个规则声明都由一系列用等号（=）分隔的键值对构成，其中每一行都有一个关键字和一个值，例如：

```
type=Single
```

可以将多个规则保存在一个文件中，每个规则之间用空行分开。

■注：可以使用反斜杠（\）把在一行中没有写完的键值对延续到下一行，也可以使用磅值符号（#）将一行规则注释掉。

SEC 的工作非常简单。它对使用 -input 选项指定的输入文件进行监控并实时查看新的日志消息。当收到一个新消息时，它就扫描规则列表看有没有匹配新消息的规则。如果找到了与新消息相匹配的规则，它就按照规则指定的操作运行，例如，将消息发送给系统管理员，运行一个脚本，或其他类似的操作。如果没有匹配的规则，SEC 就会忽略此消息，什么也不做，继续等待下一个新消息。

■注：SEC 只是会读出日志信息的内容，它并不会对其中的日志条目进行更改或删除操作。

创建 SEC 规则

再来看一个规则声明的例子，以便能更好地理解 SEC 和 SEC 规则是如何工作的。示例规则中将会规定，获取一条 syslog 消息并在收到时使用 SEC 发出警告，当使用 su 命令切换到 root 用户时就会触发该消息。因为没有人可以使用 root 用户，所以希望针对该消息发出警告，负责的管理员只使用 sudo 命令来执行一些需要管理员权限的操作。以下是此消息的一个示例。

```
Feb 15 22:29:34 au-mel-rhel-1 su: pam_unix(su-l:session): session opened for user ➡  
root by jsmith(uid=500)
```

此消息在 Red Hat 上被记录到文件 /var/log/secure 中，在 Ubuntu 上被记录到文件 /var/log/auth.log 中。因此，在相应的发行版本中都需要使用 -input 选项配置 SEC 分别对这两个文件进行监控。在 Red Hat 中的配置为：

```
-input /var/log/secure
```

在 Ubuntu 中的配置为：

```
-input /var/log/auth.log
```

首先，创建一个文件来保存这些新规则，这就是位于 /etc/sec 目录下的 security.sec 文件。当 SEC 启动时，此文件及其中包含的规则将会被调用。

```
$ sudo touch /etc/sec/security.sec
```

然后在文件中创建一条可以匹配上述消息的规则，并能在匹配时输出所需信息。示例配置如列表 18-12 所示。

列表 18-12 SEC 规则声明示例

```
type=Single  
continue=TakeNext  
ptype=regex  
pattern=session opened for user root by (\w+)  
desc=Root user login by $1 - is this authorised?  
action=shellcmd /bin/echo '$0' | /bin/mail -s "%s" root@example.com
```

下面逐行地对此例进行讨论。第一行指定了正在使用的 SEC 规则类型。在列表 18-12 中使用的是最简单的类型的规则，叫做 Single，此种类型的规则只是简单地找到一个消息并执行一个固定的操作。

■注：还有其他一些不同的类型，本章也会加以讨论。

列表 18-12 中的第二行是可选的。continue 行有 3 个可能的选项：TakeNext，DontCont，和 GoTo。第一个选项“TakeNext”告诉 SEC 即使日志条目匹配了规则，它也要继续搜索整个文件，看看还有没有其他匹配规则的条目。

第二个选项“DontCont”告诉 SEC 如果日志条目匹配了规则，它就会停止在那里，也不要再尝试将条目匹配到其他规则，这就意味着将会对照配置文件中的每一个规则对条目进行

检查，直到匹配到一个设置为 DontCont 的 continue 规则为止。这对于一些消息可能会与配置文件中的多条规则相匹配的情况会是非常有用的。这种情况的一个例子是一条消息有多个含义或多个目的。例如，用户登录消息可用于进行用户登录记录统计，同时如果登录的是 root 用户，则还可以将此消息以 E-mail 的方式发送给管理员。可以使用一个设置为 TakeNext 的 continue 选项来进行用户统计。处理完这个规则后，还要将消息与配置文件中的其他规则进行匹配，接着匹配到将 root 用户的登录消息 E-mail 给管理员的规则。

最后一个选项 “GoTo” 告诉 SEC 将日志条目传给当前配置文件中肯定存在的一条特定规则。

■注：如果在规则声明时忽略 continue 选项，则 SEC 默认设置 continue 的值为 DontCont，这样当一个规则与一条日志条目相匹配时，SEC 为停止。

规则声明中接下来的两行设置 SEC 对特殊的事件进行匹配。第一个是 ptype，就是模式类型。模式类型告诉 SEC 如何解析下一行，即 pattern 行中定义的信息。可以使用表 18-5 中所示的模式类型。

表 18-5 SEC 模式类型	
模式类型	解释
RegExp[number]	指定一个 Perl 正则表达式
SubStr[number]	指定一个子字符串
PerlFunc	调用一个 Perl 功能
NRegExp[number]	输出一个否定的正则表达式，模式匹配的结果为否定
NSubStr[number]	输出一个否定子字符串，模式匹配的结果为否定
NPerlFunc	调用一个 Perl 功能，结果为否定

■注：模式类型后面的 number 部分告诉 SEC 对规则和最后一条日志条目的数字进行比较。如果 number 留空，SEC 默认其最后收到的日志条目数字为 1。

下面主要讨论一下 SubStrate 和 RegExp 两种模式类型。SubStrate 模式类型匹配字符串数据，例如：

```
ptype=SubStr
pattern= dhclient: No DHCPOFFERS received.
```

此设置将匹配消息 “dhclient:No DHCPOFFERS received”，同时不使用任何正则表达式。

列表 18-12 使用标准的 regexp 模式类型，此模式类型告诉 SEC 将 pattern 行解释为 Perl 正则表达式。

■注：其他模式类型，参见 SEC 的 man 页。

列表 18-12 中的第 4 行（patrern 行）是模式自身。示例中此行是一个正则表达式，此正则表达式规定要对含有字符串 “session opened for user by (\w+)” 的消息进行匹配。

(\w+) 是一个正则表达式，它告诉 SEC 捕获任何在上述字符串中 by 后的字符。此例中，规则将匹配以 root 用户身份打开会话连接的用户名。

上面把正则表达式的部分 “[\w+]” 放在了括号中。在一个正则表达式里，pattern 行置于括号中的任何内容都将成为可以被 SEC 使用的变量。

此例中，(\w+) 就变成了变量\$1，所有后续放在括号中的数据将变成\$2、\$3 等。因此，如果针对此规则的被测消息如下所示。

```
su: pam_unix(su-l:session): session opened for user root by jsmith(uid=500)
```

则此消息将被匹配，而变量\$1 也就被赋予了一个值，即 jsmith。

■提示：另一个特殊的变量 “\$0” 是为日志条目内容或正在此时的规则预留的。此例中，变量\$0 就保存了输入行 “su: pam_unix(su-l:session):session opened for user root by jsmith(uid = 500)” 的内容。

列表 18-12 中的第 5 行是 desc 键值对，这是对匹配事件的文字描述。描述中可以是用在模式中定义的任何变量，比如上面在 pattern 行中定义的\$1 变量。

因此，列表 18-12 中的 desc 描述可以设置为 “Root user login by \$1 - is this authorized?.”。若使用前面段落中提到过的消息数据进行测试，则结果是此描述变为 “Root user login by jsmith - is this authorized?”。应该注意到，这里的 dec 定义中使用了 pattern 行中定义的变量\$1。

■提示：在 SEC 中还有一种终极的机构化的描述变量，那就是%s 变量。

列表 18-12 中的第 5 行和最后一行是 action 键值对，此行告诉 SEC 当规则匹配到了一个到来的消息应该作何处理。

在 action 行中，除了在模式中定义的所有变量（变量%0 以及标识 desc 行的变量%s），还可以使用另外两个变量：%t（一个与 date 命令的结果相等的时间戳）以及%u（一个与 time 命令结果相等的数字化的时间戳）。

■注：SEC 认为一个空行或注释就是当前规则声明的结束，因此只有在规则声明开始或结束的时候才能使用注释行或空行。

到此为止，已经了解了一个简单的 SEC 规则，这个规则只是大体上描述了 SEC 的能力。

下面再通过一个例子来看一下 SEC 还能做什么。列表 18-13 中使用了规则类型 SingleWithThreshold 来识别多次失败的 sshd 登录尝试。

列表 18-13 使用规则类型 SingleWithThreshold

```
type=SingleWithThreshold
ptype=regex
pattern=(\w+)\s+sshd\[([d+])\]:\s+Failed password for (\w+) from ([d+].[d+].[d+].[d+])
port [d+]\w+[d+]
desc=User $2 logging in from IP $3 to host $1 failed to enter the correct password
thresh=3
window=60
action=write /var/log/badpassword.log %s
```


希望可以使用此规则对以下日志条目的变化进行匹配。

```
Feb 15 23:23:13 au-mel-rhel-1 sshd[738]: Failed password for jsmith from 192.168.1.10 port 44328 ssh2
```

进行这种匹配所使用的规则类型为 `SingleWithTreshold`，此类型的规则在规定的时间内对日志条目进行匹配并对匹配的次数进行计数。时间窗由 `window` 选项指定，单位为秒。在列表 18-13 中，时间窗设为 60 秒。SEC 规则第一次匹配一条消息后，时间窗就开始计时。

接下来，SEC 把匹配到的次数与限定匹配次数值进行比较，此限定值在列表 18-13 中使用 `thresh` 选项被设定为 3 次。若匹配的次数在时间窗限定的时间之内达到了限定次数，则运行 `action` 行所设定的操作。

在列表 18-13 中，指定的操作是使用 `write` 操作把 `desc` 行指定的内容写入一个特定的文件 `/var/log/sshdbruteforce.log` 中。`write` 操作可以将内容写入文件、命名管道或标准输出设备。

提示：也可以在这里执行的另一个操作是添加一个防火墙规则组织此 IP 地址的 SSH 访问。

是不是还有其他可用的规则类型呢？答案是肯定的，SEC 为其复杂时间关联能力准备了大量可用规则。表 18-6 列出了其他可用的规则类型。

表 18-6 SEC 规则类型	
规则类型	描述
SingleWithScript	匹配一个事件后，执行一段脚本，然后根据脚本的输出值进行下一步操作
SingleWithSuppress	匹配一个事件后，立即执行一个操作，然后在 x 秒内忽略其他事件匹配
Pair	成对匹配设置。当匹配到一个初始事件时，立即执行一个操作。忽略任何后续事件匹配直到找到匹配的成对事件，此时再执行另一个操作
PairWithWindow	也是成对匹配设置。当匹配到一个初始事件时，等待 x 秒，期待成对事件的到来。若成对事件在指定的时间窗内到来，执行一个操作若成对事件在指定的时间窗内没有到来，执行另一个操作
SingleWithThreshold	在 x1 秒时间内对匹配到的事件数量进行计数，若多于一个门限值的 t1 事件发生，执行一个操作。此 x1 时间内的任何其他事件都将被忽略
SingleWith2Thresholds	在 x1 秒时间内对匹配到的事件数量进行计数，若多于一个门限值的 t1 事件发生，执行一个操作。接下来再次开始计数匹配到的事件数量，若在 x2 秒时间内计数少于一个门限值 t2，则执行另一个操作
Suppress	抑制任何事件匹配。可以用它来排除任何与后面的规则相匹配的事件。这对于避免高数据量、低内容信息对 SEC 的阻塞非常有用
Jump	将匹配事件提交到另一个规则集
Options	为规则集设置选项
Calendar	在指定的时间执行一个操作

那么如何使用这些规则类型呢？下面看一个附加的示例。列表 18-4 专门展示了 `Pair` 规则类型的使用。

列表 18-14 使用 Pair 规则类型

```
type=Pair
ptype=regex
pattern=(\w+\s+\d+\s+\d\d:\d\d:\d\d)\s+(\w+)\s+su\ (pam_unix\)(\[\d+\])\:\s+session ➡
opened for user root by (\w+)\s+(\w+=\d+)
desc=User $4 has succeeded in an su to root at $1 on system $2. Do you trust ➡
user $4?
action=shellcmd /bin/echo '%s' | /bin/mail -s "SU Session Open Warning" ➡
root@example.com
ptype2=regex
pattern2=(\w+\s+\d+\s+\d\d:\d\d:\d\d)\s+$2\s+su\ (pam_unix\)$3\:\s+session closed ➡
for user root
desc2=Potentially mischievous user %4 has closed their su session at %1 on system %2
action2=shellcmd /bin/echo '%s' | /bin/mail -s "SU Session Close Warning" ➡
root@example.com
```

上例中，使用 Pair 实时监测是否有人在系统中使用 su 命令切换到 root 用户，并对日志文件进行监控以确定此人何时关闭 su 会话。下面试着对下述两条日志条目的变化进行匹配。

```
Feb 15 22:29:34 au-mel-rhel-1 su(pam_unix)[17354]: session opened for ➡
user root by jsmith(uid=500)
Feb 15 23:56:45 au-mel-rhel-1 su(pam_unix)[17354]: session closed for user root
```

使用的规则类型为专门设计用来检测日志条目匹配对的 Pair 规则，也可以使用 PairWithWindow 规则类型，此规则用来在一个特定的时间窗内找到日志条目的匹配对，很像在列表 18-13 中的 SinigleWithThreshold 规则类型。

使用 Pair 规则类型，实际上定义了两组模式类型、模式、说明以及操作项目。因为此规则要匹配两个日志条目。第二组项目定义时都带有数字 2 后缀，成为 ptype2、pattern2 等，以示与第一组的区别。

第一组项目用于匹配第一个日志条目，比如当匹配到日志条目时，执行 action 行中定义的操作。第二组条目用于匹配第二组日志条目，比如当匹配到第二个日志条目时，执行 action2 行中定义的操作。

对于第一组模式类型和模式，使用了一个正则表达式模式类型。模式中定义了一些将要进行匹配的日志条目元素变量，即发生 su 对话的主机名、使用 su 命令的用户、对话开启与关闭的时间以及分配给 su 命令的进程 ID。

稍后将会看到在 desc 和 action 行中对这些变量的使用。在列表 18-14 中对 action 行的定义为 shellcmd，即当匹配到日志条目时，执行一个 shell 命令。

第二组模式类型也是一个正则表达式。那么在此模式中，如何才能确定标识 su 对话结束的日志条目和最初打开 su 对话的日志条目是有关联的呢？对于这个问题，SEC 可以使用第一个模式行 pattern 中的变量，而这些变量则可以作为在第二个模式行 pattern2 中正则匹配表达式的一部分。

在第一个 pattern 行中定义了 su 对话发生的系统主机名为变量\$2，会话进程 ID 为变量\$3。如果在 pattern2 行中使用这些变量，SEC 就会知道使用的是在第一个 pattern 行中定义的变量。因此，就可以使用主机名和进程 ID 对传入的日志条目和第一次匹配到的日志条目进行匹配。

但是这样做又会出现其他的问题。例如，当在 desc2 行中使用变量时，SEC 如何区分所

使用的变量是在哪一个模式行中定义的呢？如果想在 desc2 或 action2 行中使用第一个 pattern 行中定义的变量，则变量的前缀使用%，而第二个 pattern 行中定义的变量的前缀为\$。可以看到，第一个 pattern 行中定义的变量\$4，在 desc2 中使用时变成了%4。

另一个有用的规则类型是 Suppress。列表 18-15 是 Suppress 规则的一个示例。

列表 18-15 使用 Suppress 规则类型

```
type=Suppress
ptype=regexp
pattern=\w+\s+syslogd\[\d+\]:\s+STATS: dropped \d+
```

列表 18-15 中的规则用来对如下日志条目进行抑制。

```
Feb 15 21:19:23 syslogd[22565]: STATS: dropped 0
```

Suppress 规则类型很简单，只包含一个规则类型、一个模式类型以及一个匹配模式。事件抑制对于停止 SEC 对已知无价值事件的处理是非常有用的。可以在配置文件的开始定义一系列 Suppress 规则以停止 SEC 对不重要信息的处理。要小心设置，确保没有抑制有用信息，而且要特别小心不要让正则表达式过于宽泛，以至于抑制了需要看到的消息。

在 Suppress 规则中也可以使用 Substr 模式类型。使用子字符串而不是正则表达式重写列表 18-15，如下所示。

```
type=Suppress
ptype=substr
pattern=This message is to be suppressed.
```

要使用子字符串规则匹配日志条目，pattern 行中的内容必须和要匹配的日志条目内容完全一致。如果有必要，还可以使用反斜杠结构\t、\n、\r 或\s 来分别表示制表符、换行符、回车符或空格符。

提示：由于在 Perl 中使用反斜杠来表示特殊字符，如果在子字符串中或正则表达式中需要输入反斜杠，也必须进行转换。如在 Perl 中\\表示一个反斜杠。

Suppress 不是唯一一个可以进行消息抑制的规则类型，还可以使用 SingleWithSuppress 规则类型。此规则类型设计用来匹配单个日志条目，执行一个操作，然后在使用 window 行定义的一个固定时间内抑制匹配同一条规则的其他日志条目。

此规则用来允许进行消息压缩。在同一条日志产生了多个实例、但只需在首次匹配时通知管理员或执行某个操作就可以的场合，消息压缩是很有用的。可以将 100 条消息压缩到一次响应或一次操作，而不是对每一条消息产生 100 个不同的响应或操作。

列表 18-16 中是 SingleWithSuppress 规则类型的一个示例。

列表 18-16 使用 SingleWithSuppress 规则类型

```
type=SingleWithSuppress
ptype=RegExp
pattern=(\S+): Table overflow [0-9]+ of [0-9]+ in Table (\S+)
desc=Please check for a table overflow in $2
action=shellcmd notify.sh "%s"
window=600
```


列表 18-16 使用一个正则表达式检测数据库产生的表格溢出消息。大家都知道，此消息可能在很短的时间内发生很多次，因此可以使用 `SingleWithSuppress` 规则只匹配第一条日志条目并将出错信息通知到相关用户。

若在随后的 600 秒内（使用 `window` 行定义）还匹配到了额外的日志条目，则不执行任何操作。如果在第一条日志条目匹配后超过了 600 秒时间，日志条目在此出现，则又执行一次操作，此后其他匹配到的日志条目将又会被抑制 600 秒时间。举例来讲，这种情况可能会发生在起初的问题没有得到解决并需要再次通知的场合。

在上述的几个示例中，只讨论了 SEC 的两个操作，即 `write` 和 `shellcmd`。在 SEC 中还有其他可用的操作。表 18-7 中列出了主要的几个，可以在 SEC 的 `man` 页面中找到其他操作。

表 18-7 SEC 操作	
操作	描述
<code>assign %letter [text]</code>	将 <code>text</code> 中的内容赋给用户定义的 <code>%letter</code> 变量。也可以在 <code>text</code> 中使用其他的 <code>%</code> 变量，就和在模式中定义的变量一样。若 <code>text</code> 中不指定任何内容，则默认使用 <code>%s</code> 变量
<code>event [time] [event text]</code>	每隔 <code>time</code> 指定的秒数，产生一次在 <code>event text</code> 中指定事件。SEC 将把 <code>event text</code> 中的字符串视为一个日志条目并将其与所有规则进行匹配。若没有指定任何 <code>event text</code> ，则默认使用 <code>%s</code> 变量。若 <code>time</code> 指定为 0 或干脆省略，则事件会立即产生
<code>logonly</code>	将事件描述记录到 SEC 日志
<code>none</code>	无动作
<code>spawn shellcmd</code>	此项与 <code>shellcmd</code> 相同，同时任何从 <code>shellcmd</code> 中产生的标准输出都会输入到 SEC 中，SEC 则将其视为一个日志条目并与规则进行匹配。通过生成一个事件 0 输出行到每一个标准输出行完成此项工作。需要注意的是，经 SEC 的预处理和锁定，已经触发的 <code>shellcmd</code> 命令不会产生大量数据或陷入死循环

可以在一行中放置多个操作，各个操作之间用分号隔开。如下行所示。

```
action=shellcmd notify.sh "%s"; write /var/log/output.log %s
```

此行中将 `shellcmd` 操作和 `write` 操作结合到了一行中。

列表 18-17 中是最后一个示例——`Calendar` 规则类型。`Calendar` 规则类型结构与其他几个不同。

列表 18-17 使用 `Calendar` 规则类型

```
type=Calendar
time=1-59 * * * *
desc=This is an important message SEC needs to check
action=shellcmd purge.sh
```

`Calender` 规则类型使用了一个叫 `time` 的特殊行。`Time` 行使用由空格分隔的 `crontab` 标准 5 段格式，5 段分别是分钟、小时、一个月中的天数、一年中的月数和周几。可以使用 `Calerdar` 规则类型安排事件日程或启动日志相关的进程。通常使用 `Calender` 事件清除或管理在日志进程执行过程中使用到的文件或初始化日志报告。

18.2.5 SEC 排错

本章中的示例应该已经介绍了开始编写 SEC 规则的基本知识。更多关于 SEC 规则编写的帮助和信息，请分别参考 SEC FAQ 和网页“<http://kodu.neti.ee/~risto/sec/FAQ.html>”或“<http://kodu.neti.ee/~risto/sec/examples.html>”。

在 Red Hat 软件包中也有一些可用的示例规则，这些示例都被安装到了 `/etc/sec/examples` 目录下。

这些示例使用 Calender 类型创建了一个显示特定日志条目的规则报告。本书的源代码中也包含了这些示例，可以从查看 `001_init.sec` 文件开始。

前面也提到过，在链接“<http://www.estpak.ee/~risto/sec/#mailinglist>”处的 SEC 邮件列表也是非常好的帮助信息资源。

18.3 监 控

一旦所有的应用程序和服务都运行起来了，就需要有一些可用的机制来对其进行监控。这样做可以确保当某些重要的时间发生时主机会发出警告信息，比如主机磁盘空间满或某个服务非正常停止等。

在 IT 世界中，监控机制称为企业监管（enterprise monitoring）。就像在本书中介绍过的其他应用程序和工具一样，也有很多可用的开源工具可以执行监控的任务，如下所示。

- Hyperic (<http://www.hyperic.com/>)。
- M/Monit (<http://mmonit.com/>)。
- Nagios (<http://www.nagios.org>)。
- OpenNMS (<http://www.opennms.org/>)。
- Zenoss (<http://zenoss.com/>)。
- Zabbix (<http://www.zabbix.com/>)。

可能 Nagios（其读音可以在以下网页听到“<http://community.nagios.org/2007/02/20/nagios-pronunciation/comment-page-1/>”）是其中最著名的，下面将对它进行详细介绍。

18.3.1 Nagios 简介

Nagios 是一款流行的 GPL 许可的工具，使用此工具可以监控基础设施、应用程序甚至诸如电力和空调等的环境特征。它带有一个简单的 Web 控制台，此控制台页面提供了一个主机和服务状态的可视化窗口。图 18-1 是控制台页面的一个示例。

下面来介绍 Nagios 以及如何使用它对主机和服务进行监控，其中包括如何为主机和本书前面介绍过的一些服务设置基本的监控。

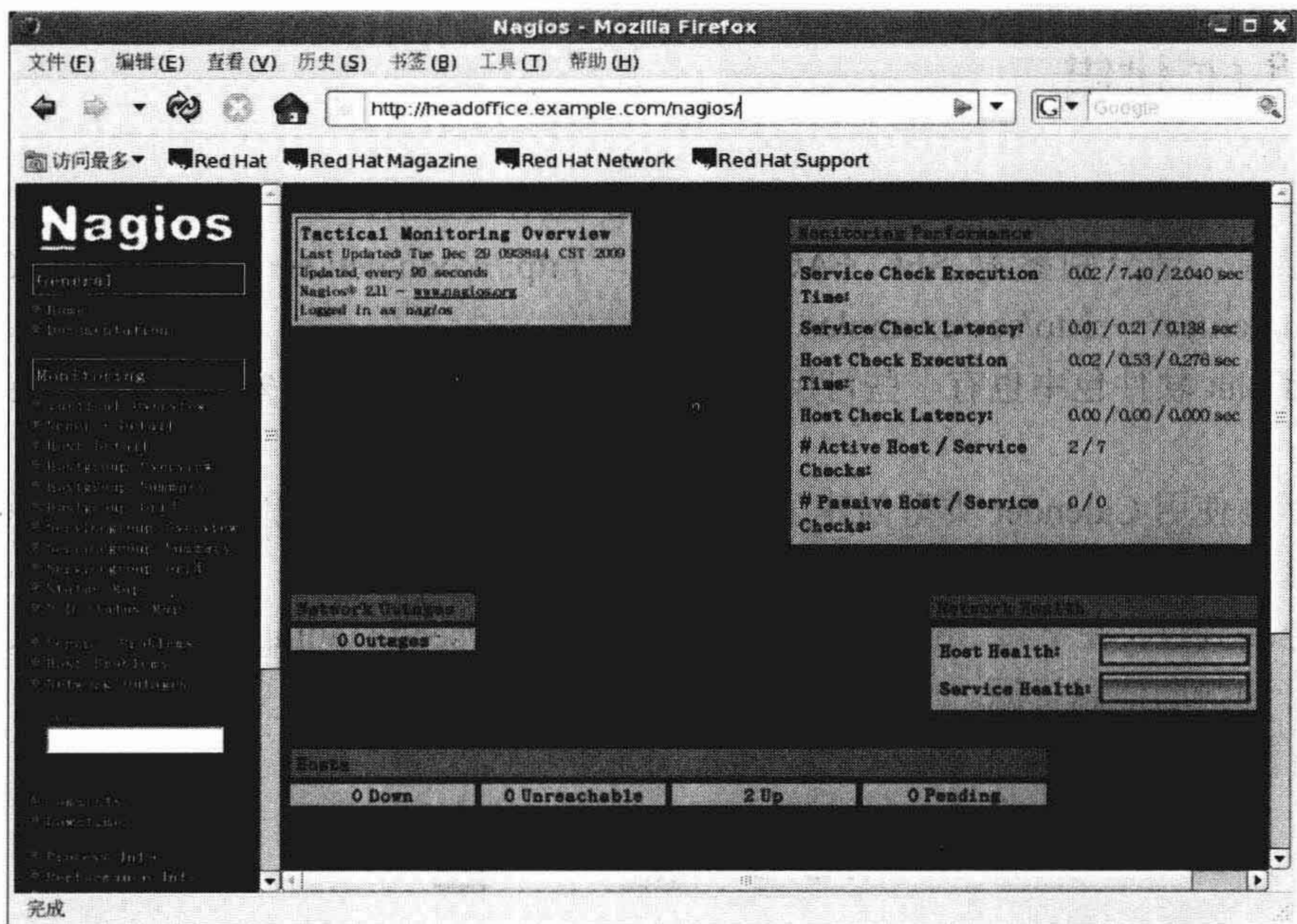


图 18-1 示例控制台页面

本书中，使用版本号为 2 的 Nagios，此版本包含了完整的特性和功能。实际上，Nagios 很复杂，一章的内容是不可能介绍完全部内容的。幸运的是，Nagios 的文档十分完备，可以在以下链接处找到第 2 版的文档，见“http://nagios.sourceforge.net/docs/2_0/”（第 3 版的文档在 http://nagios.sourceforge.net/docs/3_0/）。

■注：有很多书都讨论了 Nagios 的配置，其中就包括本书的作者之一——James Tunrbull 编写的《精通 Nagios 2.0》（Pro Nagios 2.0）。还可以在以下链接处看到其他的书籍，见“<http://www.nagios.org/propaganda/books/>”。

Nagios 具有可以在各种平台上检测多种不同主机和服务的能力，其中包括 Linux、BSD 甚至是 Windows（Windows 监控命令请参考 http://nagios.sourceforge.net/docs/3_0/monitoring-windows.html）。

Nagios 作为守护进程运行，并对主机上正在运行的服务及其状态进行监控。它即可以通过监控确认服务是否在运行，又可以确定服务运行是否正常。比如，如果对一个正使用 ICMP ping 命令（此命令在第 6 章介绍过）的主机进行监控，就应该将其配置为如果不能与主机建立连接或返回响应的时间比设定值长则发出警告信息。

进行这项监控，需要在 Nagios 配置文件中将主机和服务定义为对象来告诉 Nagios 关于主机和服务的信息。每一个主机都要定义到 Nagios 中，在每个主机中运行着的每一个服务也要进行定义。

还要定义命令来告诉 Nagios 如何对主机和服务进行监控。每一个服务都要使用特定的命令来查看其运行状态，而且每一个命令都指定了用来查看服务状态的二进制代码或脚本代码。

■注：还要定义发出警告信息的命令，如当服务检测失败时发出一个 e-mail。

为了简化命令的创建，Nagios 自带了一系列插件 (plug-in)，这些插件都是设计用来检查特定服务的二进制代码，比如 Nagios 有一个名为 `check_icmp` 的插件，用来使用 ICMP 的 `ping` 命令确认主机是否处于活动状态。

Nagios 可以查询本机及远程主机上的服务，可以直接连接到主机完成查询（比如可以在 25 端口上直接连接到 SMTP 服务器并测试是否可以收到 e-mail）或者通过在主机上安装一个代理软件，并通过此代理将测试结果返回监控主机。

除了监控主机和服务，Nagios 还有其他各种有用的功能，包括以下几个方面。

升级模式。如果主机或服务不能自动回复或没有修复则允许警告升级（参见“http://nagios.sourceforge.net/docs/2_0/escalations.html”）。

指定事件处理器的能力。当主机或服务失败时将触发指定的事件处理器。事件处理器可以运行如重启服务或删除暂存文件以试图自动恢复主机或服务的任务（参见“http://nagios.sourceforge.net/docs/2_0/eventhandlers.html”）。

指定主机和服务间的父子关系和依赖关系。比如，如果一个路由不工作了，可以设定 Nagios 无需费神再去检查此路由下的主机了，因为这些主机是不可到达的（参见“http://nagios.sourceforge.net/docs/2_0/dependencies.html”）。

可以设计一个分布式的监控环境，在里面有冗余的监控服务器，或者在其中要进行分布到多个主机或位置的检查工作（参见“http://nagios.sourceforge.net/docs/2_0/distribute0d.html”和 http://nagios.sourceforge.net/docs/2_0/redundancy.html”）。

18.3.2 安装 Nagios

Nagios 的安装很简单，Red Hat 和 Ubuntu 中都存在相应的软件包。

1. 在 Red Hat 上安装 Nagios

在 Red Hat 主机上，需要安装几个软件包，下面从安装 `libtool-ltdl` 软件包开始。

```
$ sudo yum install libtool-ltdl
```

此外，还需要 `Perl-Net-SMTP`、`fping`、`nagios` 以及 `nagios-plugins` 软件包。若要获得这些软件包，先安装一个额外的库 `RPMForge`，可以通过安装一个 `RPM` 软件包来启用这个库。

```
$ wget http://packages.sw.be/rpmforge-release/rpmforge-release-0.3.6-1.el5.rf.i386.rpm
$ sudo rpm -Uvh rpmforge-release-0.3.6-1.el5.rf.i386.rpm
```

■注：可以在以下链接处找到针对各种不同发行版本的命令和下载文件，见 <https://rpmrepo.org/RPMforge/Using>

安装完库后，就可以使用 `yum` 命令来安装所需的软件包了。


```
$ sudo yum install libtool-ltdl perl-Net-SNMP fping nagios nagios-plugins
```

2. 在 Ubuntu 中安装 Nagios

在 Ubuntu 中，只需安装 nagios2 和 nagios-plugins 两个软件包。

```
$ sudo apt-get install nagios2 nagios-plugins
```

此命令将会安装第 2 版的 Nagios 以及包括 Nagios 插件在内的所有支持软件包。

■提示：在 Ubuntu 8.10 中，下一版的 Nagios——第 3 版，也被打包在其中了。可以安装 nagios3 软件包使用第 3 版的 Nagios。Nagios 3 中有一部分升级和新特性，与此相关的内容可以参考“http://nagios.sourceforge.net/docs/3_0/whatsnew.html”。

18.3.3 启动 Nagios

在 Red Hat 和 Ubuntu 中，都是用标准的 init 脚本启动和停止 Nagios 的。在 Red Hat 上，此脚本名为 nagios。

```
$ sudo service /etc/init.d/nagios start
```

nagios 守护进程将日志记录到/var/log/nagios/nagios.log 文件中。通过查看此文件，可以确定守护进程是否成功启动，如果没有，在此文件中将会有错误信息。

在 Ubuntu 中，此脚本名为 nagios2。

```
$ sudo invoke-rc.d nagios2 start
```

Ubuntu 中，守护进程将日志记录到/var/log/nagios2/nagios.log 文件中。

18.3.4 Nagios 配置

下面快速浏览一下如何配置 Nagios。简单来讲，配置 Nagios 的步骤如下所示。

1. 为要监控的主机创建定义。
2. 为主机上要监控的服务创建定义。
3. 创建命令来监控服务。
4. 告诉 Nagios 何时进行监控。
5. 告诉 Nagios 如果检测到错误通知哪个用户。
6. 告诉 Nagios 当检测到错误时，使用什么方式进行通知，如 e-mail、IM、SMS 等。

Nagios 配置由对象构成。定义希望监控的主机为主机对象，每一个需要监控的服务都定义为一个服务对象。当然还有其他各种不同类型的对象，例如表示监控周期的时间周期对象以及告诉 Nagios 当错误发生时通知谁的联系人对象。

下面将会讨论如何在 Nagios 中定义主机，以及接下来如何配置各种不同类型的服务。其中，还包含 Nagios 配置中的其他各种不同元素。

Nagios 的配置文件在 Red Hat 中保存在/etc/nagios 下，在 Ubuntu 中保存在/etc/nagios2 下。Nagios 的配置文件都带有.cfg 后缀，主要的配置文件名为 nagios.cfg。

在 Red Hat 和 Ubuntu 中都带有一些示例配置文件以帮助读者开始使用 Nagios。在 Red Hat 中，配置文件/etc/nagios/localhost.cfg 中包含了本地主机及主机上服务的一些基本配置。在 Ubuntu 中，目录/etc/nagios2/conf.d 和/etc/nagios-plugins/config 下有更广泛的示例配置集。

Nagios 还有一种模式，可以允许在运行守护进程之前检查配置文件中是否存在错误，这对于确保配置文件中没有任何错误信息非常有用。

在 Red Hat 中，使用下面的命令进行配置文件检查。

```
$ sudo nagios -v /etc/nagios/nagios.cfg
```

在 Ubuntu 中则为以下命令。

```
$ sudo nagios2 -v /etc/nagios2/nagios.cfg
```

-v 选项检查是否所有配置都是正确的，如果是，则输出带有定义的配置对象数量的统计报告。

■注：Nagios 配置文件改变后，需要重启守护进程以对新的配置进行解析。

1. nagios.cfg 文件

nagios.cfg 配置文件包含了 Nagios 安装的基本配置。此文件中的每一个选项都是一个选项-值对。例如，Nagios 日志文件的位置使用 log_file 选项进行定义，在 Red Hat 中，如下进行定义：

```
log_file=/var/log/nagios/nagios.log
```

通常这是 nagios.cfg 文件中的第一个选项，紧随其后的分别是指定对象配置文件的 cfg_file 和 cfg_dir 选项。cfg_file 选项允许指定包含 Nagios 对象配置的一个单独文件，例如：

```
cfg_file=/etc/nagios/localhost.cfg
```

可以指定多行，事实上，很多人为了便于管理而把每一个对象类型都定义在一个单独的文件中。

```
cfg_file=/etc/nagios/hosts.cfg
cfg_file=/etc/nagios/services.cfg
cfg_file=/etc/nagios/contacts.cfg
...
```

■注：把配置文件放到一个类似于 Subversion 的版本管理系统（VCS）中是个好主意，这样的系统对文件及对文件的更改进行跟踪。通常由程序员用来跟踪源代码，现在也越来越多地被系统管理员用来跟踪配置文件了。关于版本控制的内容请参考“http://en.wikipedia.org/wiki/Revision_control”，关于 Subversion 请参考“<http://svnbook.red-bean.com/>”。

cfg_dir 选项指定一个目录。Nagios 将会加载此目录下所有后缀名为.cfg 的文件，例如：

cfg_dir=/etc/nagios2/conf.d

nagios.cfg 文件包含了许多其他有用的选项，其中的一些列在了表 18-8 中。

表 18-8 nagios.cfg 配置文件选项

选项	说明
resource_file	一个单独的配置文件，用于保存诸如路径、密码等的系统变量
nagios_user	运行 nagios 的用户名，默认为 nagios
nagios_group	运行 nagios 的组名，默认为 nagios
log_rotation_method	何时轮换日志文件。可用值为：n，不轮换；h，每小时一次；d，每天一次；w，每周一次；以及 m，每月一次
log_archive_path	用来保存归档、轮换的日志文件的目录
use_syslog	是否将 Nagios 输出记录到系统日志。默认为 1，记入系统日志，设为 0 时不记入系统日志

可以在 nagios.cfg 配置文件中对启用或禁止主机和服务检查以及示警消息发送进行全局级别的设置。可用选项的完整列表，请参考“http://nagios.sourceforge.net/docs/2_0/configmain.html”。

2. 主机配置

下面打开 Red Hat 中的/etc/nagios/localhost.cfg 文件，仔细研读一下 Nagios 的配置，并从一个主机对象的定义开始研究其中的各部分内容。

■注：在本章的源代码中包含了 Red Hat 和 Ubuntu 中的示例配置文件。

下面就从文件中的主机对象定义开始，如列表 18-18 所示。

列表 18-18 一个主机对象

```
define host{
    use          linux-server          ; Name of host template to use
    ; This host definition will inherit all variables that are defined
    ; in (or inherited by) the linux-server host template definition.
    host_name     localhost
    alias         localhost
    address       127.0.0.1
}
```

可以看到，一个对象的定义从 define 开始，在这里定义对象的类型，示例中的对象类型是主机对象，接下来是包含在大括号{}中的定义内容。定义的内容是由一系列键值对声明组成的对象属性，各个属性之间用空格隔开。示例中定义的对象有 4 个属性：use，host_name，alias 和 address。

■提示：特定对象定义的某些属性是强制性的，意味着必须定义此属性并为其指定一个值。在 Nagios 文档中，这些值都用红色标示了出来，见“http://nagios.sourceforge.net/docs/2_0/xodtemplate.html”。

use 属性告诉主机对象引用一个模板。模板是 Nagios 用来使用可能在很多对象中相同的值来填充对象定义的技术。比如，很多主机对象可能会拥有很多相同的属性和特征，与其在每一个主机对象定义中一个一个地指定每一个属性的值，还不如使用模板。于是 Nagios 使用所有在主机定义中的属性加上模板中的属性来共同创建主机对象。稍后将花一点时间来看一下在 linux-server 模板中定义的属性，如列表 18-19 所示。

此例中，主机对象其余的属性定义了其区别于其他主机的特性。host_name 属性定义了主机对象的名称，此名称必须是唯一的。只能有一个主机名为 localhost 或 headoffice.example.com。Nagios 还将 host_name 作为一个名为\$HOSTNAME\$的宏存放。

■注：使用宏，Nagios 就可以在其他对象定义中嵌入主机和服务的信息，特别是 Nagios 用来检测服务和发送通知的一些命令。在本章的后面将会有更多的宏，同时，可以在以下链接处找到所有宏的完全列表，见 http://nagios.sourceforge.net/docs/2_0/macros.html。

alias 属性指的是对象的另外一个名字，此例中使用了 localhost 作为主机的别名。此别名通常用来作为对主机较长的介绍，它也有一个相对应的宏\$HOSTALIAS\$。

最后一个属性（address）提供了主机的 IP 地址，此例监控的是本地主机，即 127.0.0.1。此 IP 地址对 Nagios 必须是可连接的，以便于提供监控服务。此属性也有一个对应的宏\$HOSTADDRESS\$。

■注：还可以为主机指定一个完全合乎要求的域名，但是指定域名需要 DNS 服务器工作正常并能解析主机名。如果 DNS 失败，则 Nagios 检测也会失败。因此，还是建议使用 IP 地址作为此属性的值。

下面来看一下在 linux-server 模板中还提供了哪些属性。列表 18-19 中就是 linux-server 主机对象模板的内容。

列表 18-19 主机对象模板

```
define host{
    name          linux-server      ; The name of this host ➡
    template
    use           generic-host      ; This template inherits ➡
other values from the generic-host template
    check_period  24x7              ; By default, Linux hosts ➡
are checked round the clock
    max_check_attempts 10          ; Check each Linux host 10 ➡
times (max)
    check_command check-host-alive ; Default command to check➡
Linux hosts
    notification_period workhours   ; Linux admins hate to be ➡
woken up, so we only notify during the day
                                   ; Note that the ➡
notification_period variable is being overridden from
                                   ; the value that is ➡
inherited from the generic-host template!
    notification_interval 120       ; Resend notification every➡
2 hours
```



```
notification_options    d,u,r          ; Only send notifications ➡
for specific host states
contact_groups          admins          ; Notifications ➡
get sent to the admins by default
register                0              ; DONT REGISTER THIS ➡
DEFINITION - ITS NOT A REAL HOST, JUST A TEMPLATE!
}
```

可以看到，在模板中定义了很多的属性。首先，定义了此模板是适用于何种对象的，此例中是适用于主机对象的。接下来，使用 `name` 属性为模板指定了一个必须唯一的名称，不能同时存在两个名为 `linux-server` 的模板。

下面一个属性是前面见到过的“`use`”，它用来指定此模板继承自何处。有点乱？简单来讲，Nagios 允许将模板链接在一起。基于这种机制可以建立相当复杂的模板模型，以尽量少的必须键入内容定义一个监控环境。稍后也会简单地看一下 `generic-host` 模板。

提示：如果在多个模板中定义了同一个属性时怎么办？应该使用哪一个呢？Nagios 使用向下继承机制，即使用对属性最后一次引用时的值。例如，如果属性 `check_period` 在模板 `generic-host` 和 `linux-server` 以及本地主机的主机对象中都进行了定义，则使用在 `localhost` 对象中定义的值。

列表 18-19 中，接下来的 3 个属性 `check_period`、`max_check_attempts` 和 `check_command` 都是相关联的。

第一个属性，`check_period` 告诉 Nagios 何时对主机进行检测。此例中，指定的时间周期是 24×7 。也需要在 Nagios 配置中对时间周期进行定义。

```
define timeperiod{
    timeperiod_name 24x7
    alias           24 Hours A Day, 7 Days A Week
    sunday          00:00-24:00
    monday          00:00-24:00
    tuesday         00:00-24:00
    wednesday       00:00-24:00
    thursday        00:00-24:00
    friday          00:00-24:00
    saturday        00:00-24:00
}
```

这是一个时间周期定义示例，其中有一个名为 24×7 的 `timeperiod_name` 属性，以及一个 `alias` 描述属性。接下来定义了一周中的每一天以及时间周期包括了这些天的哪个时间段。在此时间周期示例定义中，指定了时间周期为一个周的每一天，即每天 24 小时。

若不想包括某一天，不要定义它就好了。时间在这里定义为 24 小时，也可以指定多个时间段，例如：

```
sunday 00:00-02:00,17:00-19:00
```

这样就定义了时间周期为周日的午夜到上午 2 点以及下午 5 点到 7 点。

时间周期在 Nagios 的很多地方都有应用，但其最通常的用途是指定何时检测主机和服务以及何时发送通知信息（当主机和服务失败或其所需状态发生改变时生成此信息）。

下一个属性 `max_check_attempts` 指定了 Nagios 确认主机或服务发生问题之前对其进行检测的次数。

最后一个属性 `check_command` 告诉 Nagios 使用什么命令来检测主机状态，在这里是 `check-host-alive`。

前面已经讨论过此命令。下面再来看一下。

```
define command{
    command_name check-host-alive
    command_line $USER1$/check_ping -H $HOSTADDRESS$ -w 3000.0,80% ➡
    -c 5000.0,100% -p 1
}
```

命令的定义与其他对象一样。使用 `command_name` 属性定义命令的名称，真正要运行的命令使用 `command_line` 属性指定。此例中指定的命令如下所示

```
$USER1$/check_ping -H $HOSTADDRESS$ -w 3000.0,80% -c 5000.0,100% -p 1
```

命令的第一部分 “`$USER1$`” 是另一个 Nagios 宏。“`$USERx$`” 宏是配置变量，通常在文件 `resource.cfg`（或是文件 `nagios.cfg` 中使用 `resource_file` 配置选项定义的另外一个文件）中使用。此例中，宏 “`$USER1$`” 表示的是包含 Nagios 插件的目录。

```
$USER1$=/usr/lib/nagios/plugins
```

命令的后面部分是此命令要使用的 Nagios 插件 `check_ping`，此插件使用 ICMP ping 来检测主机的状态。

■注：此处假设被检测主机设置为允许接收 ICMP ping 信号，此内容在第 6 章做过讨论。

此命令中，可以看到一个前面学过的包含主机 IP 地址的 Nagios 宏 “`$HOSTADDRESS$`”。无论主机何时执行检测命令，此宏都会被主机的地址所代替，这也就使得同一个命令可以被多个主机对象所使用。此地址宏被指定为命令中 “`-H`” 选项的值，即在命令中指定了被 ping 主机的地址。

■注：可以通过运行带有 `--help` 选项的命令来获得大多数 Nagios 插件的帮助信息。还可以在命令行中运行 Nagios 插件，观察其运行过程、命令行选项以及运行结果。

后面的两个选项（`-w` 和 `-c`）指定了运行此检查任务的线程。如果这些线程终止运行，Nagios 将更新主机或服务状态。

主机或服务各有不同的状态。一个主机可以有 UP、DOWN 或 UNREACHABLE 等状态，而服务则可以处于 UP、WARNING、CRITICAL 或 UNKNOWN 状态。

■注：当处于以下情况时，主机处于 UNREACHABLE 状态，即设置了依赖关系和父子关系，但其父主机或依赖的主机不存在。

插件自身只能返回 WARNING、CRITICAL 和 UNKNOWN 状态（通常当插件运行失败或返回错误信息而不是有效状态信息时设置为 UNKNOWN）。当插件在主机上运行时，Nagios

将对其返回的状态信息进行解析并转换为合适的 UP 和 DWON 状态信息，如表 18-9 所示。

表 18-9		Nagios 插件状态转换
插件状态	主机状态	状态说明
OK	UP	主机已启动
WARNING	UP or DOWN	主机可能为启动，也可能为停止，默认为启动（参见 http://nagios.sourceforge.net/docs/2_0/configmain.html#use_aggressive_host_checking ）
UNKNOWN	DOWN	主机停止
CRITICAL	DOWN	主机停止

■注：主机状态在 Nagios 中也有对应的宏定义：\$HOSTSTATE\$。

-w 和 -c 选项各取一值，即组成以毫秒为单位的 ping 消息往返时间和不中断线程所需的 ping 命令成功比例。因此，如果 -w 线程终止，则产生 WARNING 状态，Nagios 将主机标识为 UP。然而，如果 -c 线程终止，则产生 CRITICAL 状态，Nagios 将主机标识为 DOWN，随后产生一个警告消息。命令中的最后一个选项 -p 指定了发送 ping 消息的次数。

■注：插件除了返回诸如 WARNING 或 CRITICAL 的状态信息外，还会输出一些状态的描述信息，而这些信息可以用在警告信息中或在控制台上显示。例如，check_ping 插件返回信息“PING OK - Packet loss = 0%, RTA = 3.98 ms”。此输出信息也有对应的宏定义，分别是对应主机的“\$HOSTOUPUT\$”和对应服务的“\$SERVICEOUTPUT\$”。

列表 18-19 中的下一个属性是 notificaiotn_period，这个属性和检查周期有些微的不同。检查发生在 check_period 期间，在示例中就是 24 × 7 期间，而警告信息（状态发生改变时产生的警报）却只在 workhours 时间周期内发送。

```
define timeperiod{
    timeperiod_name workhours
    alias            "Normal" Working Hours
    monday          09:00-17:00
    tuesday         09:00-17:00
    wednesday       09:00-17:00
    thursday        09:00-17:00
    friday          09:00-17:00
}
```

可以看到，workhours 时间周期是上午 9 点到下午 5 点，而不是由 24 × 7 检查周期定义的一周中每天 24 小时。

下一个属性 notification_interval 配置 Nagios 如果主机的状态没有发生改变则隔多长时间重新发送一次警告信息，这里设置的是 120 分钟，也就是每隔 2 个小时一次。

Notification_options 属性定义了 Nagios 何时发送警告信息。这里设置为 d、u 和 r，意味着当主机状态为 DOWN（d）或 UNREACHABLEE（u）时，Nagios 将发送警告信息。最后一个选项 r 表示当主机状态恢复（比如，从 DOWN 或 UNREACHBALE 状态转为 UP 状态时）

时发送警告信息。

下一个属性 `contact_groups` 告诉 Nagios 当警告信息产生时将其发送给谁。此例中，此属性的值设为 `admins`，此值指向一个 `contactgroup` 对象。联络组是联系人的集合，就是当警告信息产生时要通知的所有人员，如系统用户自己或其他系统管理员。一个 `contactgroup` 对象类似如下所示。

```
define contactgroup{
    contactgroup_name    admins
    alias                Nagios Administrators
    members              nagios-admin
}
```

联络组有一个使用 `contactgroup_name` 属性定义的名称，有使用 `alias` 属性定义的描述信息，还有一个使用 `members` 属性指定的用户组中联系人的列表。此例中，联络组中的唯一成员是 `nagios-admin`，如下所示。

```
define contact{
    contact_name          nagios-admin
    alias                Nagios Admin
    service_notification_period 24x7
    host_notification_period 24x7
    service_notification_options w,u,c,r
    host_notification_options d,r
    service_notification_commands notify-by-email
    host_notification_commands host-notify-by-email
    email                nagios-admin@localhost
}
```

联系人的定义很简单。每一个联系人都使用 `contact_name` 属性指定一个名称并用 `alias` 属性进行描述。

对于每一个联系人，都要定义其何时收到警告信息以及会收到何种类型的警告信息。

使用 `service_notification_period` 和 `host_notification_period` 指定收到信息的时间。就像前面看到的一样，此例中，联系人 `nagios-admin` 将会在 24×7 的时间周期内，也就是事实上的每个周的每天 24 小时时间内，收到主机和服务的警告信息。

使用 `service_notification_options` 和 `host_notification_options` 定义联系人收到哪条警告信息。对于服务来讲，联系人 `nagios-admin` 将会收到分别由 `w`、`u` 和 `c` 选项指定的 **WARNING**、**UNKNOWN** 或 **CRITICAL** 警告信息，`r` 选项表示还可以收到主机恢复警告消息。对主机来讲，联系人将会收到 **DOWN** (`d`) 和恢复 (`r`) 警告信息。

`service_notification_commands` 和 `host_notification_commands` 属性指定了 Nagios 发送警告信息的命令。可以同时指定由逗号分隔的多条命令，对这些命令的定义就和定义用来检测主机和服务的命令一样。下面来看一下其中的一个命令 `notify-by-email`。

```
define command{
    command_name host-notify-by-email
    command_line /usr/bin/printf "%b" "***** Nagios 2.11 ➡
*****\n\nNotification Type: $NOTIFICATIONTYPE$\nHost: ➡
$HOSTNAME$\nState: $HOSTSTATE$\nAddress: $HOSTADDRESS$\nInfo: ➡
```



```
$HOSTOUTPUT$\n\nDate/Time: $LONGDATETIME$\n" | /bin/mail -s ➡  
"Host $HOSTSTATE$ alert for $HOSTNAME$!" $CONTACTEMAIL$  
}
```

和前面看到的命令一样，这里也使用 `command_name` 属性指定了命令的名称，并使用 `command_line` 属性指定了实际要运行的命令。此例中指定的命令将把包括一些宏在内的许多文字信息输出到 `/bin/mail` 二进制文件中。

此配置将给每一个需要知道的联系人发送一个 e-mail 以通知状态发生了改变。例如，如果 Nagios 正在监控主机 `gateway.example.com` 时，一项检测出出了问题，那么将会产生一个类似于如下所示的警告信息。

```
***** Nagios 2.11 *****  
  
Notification Type: PROBLEM  
Host: gateway.example.com  
State: DOWN  
Address: 192.168.0.254  
Info: PING CRITICAL - Packet loss = 100%  
Date/Time: Fri Feb 13 00:30:28 EST 2009
```

■提示：Nagios 不仅仅可以通过 e-mail 方式发送警告信息。事实上，Nagios 可以将警告信息发送到任何构建到警报命令中的设备，如使用 SNMP，类似于 Jabber、传呼机等即时消息或甚至是类似于 OTS 或 Trac 的售票系统。更多细节请参考如下链接处的警告信息文本，见 http://nagios.sourceforge.net/docs/2_0/notifications.html。

表 18-10 中列出了可以在警告命令中使用的宏。

表 18-10 警告命令中的宏

宏名	说明
\$NOTIFICATIONTYPE\$	警告的类型。比如，若主机有问题，其值为 PROBLEM；若主机恢复了，则其值为 RECOVERY
\$HOSTNAME\$	发出警告的主机名
\$HOSTSTATE\$	当前主机状态，如 UP 或 DOWN
\$HOSTADDRESS\$	主机 IP 地址
\$HOSTOUTPUT\$	用于检测主机状态的命令返回文本
\$LONGDATETIME\$	长时间日期格式（如 Fri Feb 13 00:30:28 EST 2009）
\$CONTACTEMAIL\$	要发送信息的联系人邮箱地址

■注：宏及宏的使用方法详细列表请参考“http://nagios.sourceforge.net/docs/2_0/macros.html”。

回到联系人定义，最后一个属性 `email` 指定了一个发送警告信息的电子邮件地址，它也有一个对应的宏“`$CONTACTEMAIL$`”。

再回到模板，可以看到列表 18-19 所示的模板中的最后一个属性是 `register`。这个属性告诉 Nagios 此文件是一个模板而不是一个真正的主机定义，当 `register` 设为 0 时，Nagios 将不会尝试创建一个主机对象，而是忽略它。`register` 的默认配置为 1，表示任何不明确指定 `register`

为 0 的对象定义都将被认为是一个真正的主机对象并用 Nagios 对其进行监控。

下面再来简单看一下父模板 generic-host，如列表 18-20 所示。

列表 18-20 generic-host 模板

```
define host{
    name generic-host ; The name of this host template
    notifications_enabled 1 ; Host notifications are enabled
    event_handler_enabled 1 ; Host event handler is enabled
    flap_detection_enabled 1 ; Flap detection is enabled
    failure_prediction_enabled 1 ; Failure prediction is enabled
    process_perf_data 1 ; Process performance data
    retain_status_information 1 ; Retain status information ➡
    across program restarts
        retain_nonstatus_information 1 ; Retain non-status ➡
    information across program restarts
        notification_period 24x7 ; Send host notifications ➡
    at any time
        register 0 ; DONT REGISTER THIS ➡
    DEFINITION - ITS NOT A REAL HOST, JUST A TEMPLATE!
}
```

■注：这里并不讨论这些选项的细节问题，可以在以下链接处找到相关的信息，见“http://nagios.sourceforge.net/docs/2_0/xodtemplate.html#host”。

读者可以模仿上述的例子来定义自己的主机。例如，gateway.example.com 主机可以如下进行定义。

```
define host{
    use linux-server ; Name of host template to use
    ; This host definition will inherit all variables that are defined
    ; in (or inherited by) the linux-server host template definition
    host_name gateway.example.com
    alias gateway.example.com
    address 192.168.0.254
}
```

■提示：不要忘了添加任何新的配置后都要重启 Nagios。

这样就为 gateway.example.com 定义了一个主机对象并指定要使用前面刚刚讨论过的模板。配置中为主机指定了一个内部 IP 地址 192.168.0.254，Nagios 将使用此地址通过 ICMP 尝试对主机进行监控。Gateway.example.com 主机上的防火墙应该允许 ICMP 数据包通过以确保监控可以进行。

■注：还有几个和主机监控有关的其他的对象定义，在这里都没有来得及讨论。这些对象定义可以允许进行主机分组，允许建立主机间的依赖关系，以及提供相似的功能性等。可以在以下链接处找到对象类型及其属性的完整列表，见“http://nagios.sourceforge.net/docs/2_0/xodtemplate.html”。

3. 服务配置

上面已经介绍了一些有关主机对象的内容，下面要讨论的是一个服务对象。服务使用服务类型对象进行逆光定义并与其底层主机相连接。例如，在现存的配置示例基础上，列表 18-12 显示了一个检查 root 分区磁盘空间的服务。

列表 18-21 一个服务定义

```
define service{
    use                local-service      ; Name of service template to use
    host_name           localhost
    service_description Root Partition
    check_command        check_local_disk!20%!10%!/
}
```

这个服务定义很简单。use 属性指定了服务使用的模板。host_name 属性指定了服务运行于何种主机上，此例中为 localhoat 主机。service_description 是对服务的描述。最后，check_command 属性指定了服务对正在监控的对象进行检测时所使用的命令。这个 check_command 稍微有些不同，因为实际运行的命令之后还有一个字符串。

!20%!10%!/

这个字符串由传递给命令定义的变量组成，每一个变量的值都有一个惊叹号(!)作为前缀。所以这里是把 20%、10%和/传递给了命令。稍后将会看到，这样的设置使此命令可以被多个服务重复使用。

简单看一下 check_local_disk 命令。

```
define command{
    command_name        check_local_disk
    command_line         $USER1$/check_disk -w $ARG1$ -c $ARG2$ -p $ARG3$
}
```

就像前面的定义一样，使用指定了宏\$USER1\$的 command_line 给出了要运行的插件的路径。此处要使用的插件是 check_disk，此插件用来检测本地硬盘的运行状态。

命令中也有-w 和-c 选项——这两个选项在前面讨论过，其作用是设置 WARNING 和 CRITICAL 状态的线程值。最后，还有-p 选项，此选项指定要监控的硬盘分区。然而，在此命令中，上述 3 个选项的值为\$ARGx\$，分别对应着\$ARG1\$、\$ARG2\$和\$ARG3\$。它们每一个都对应着列表 18-21 里 command_line 中传递的参数，代入后，实际的 command_line 看起来如下所示。

```
command_line          $USER1$/check_disk -w 20% -c 10% -p /
```

如此设置的结果就是当硬盘可用空间只有 20%时将产生 WARNING 状态，当硬盘只剩 10%空间时将产生 CRITICAL 状态，两种情况都被应用于 root 文件系统或者说根文件系统(/)。

若要创建一个对硬盘中其他分区进行监控的服务，如/var 分区，可以按如下格式进行创建。

```
define service{
    use                local-service      ; Name of service template to use
    host_name           localhost
```



```

service_description Var Partition
check_command      check_local_disk!20%!10%!/var
}

```

在讨论其他服务之前，先简单地看一下服务使用的 local-service 模板的内容。

```

define service{
    name          local-service          ; The name of this service template
    use            generic-service        ; Inherit default ➡
values from the generic-service definition
    check_period   24x7                  ; The service can be ➡
checked at any time of the day
    max_check_attempts 4                  ; Re-check the ➡
service up to 4 times in order to determine its final (hard) state
    normal_check_interval 5              ; Check the service ➡
every 5 minutes under normal conditions
    retry_check_interval 1               ; Re-check the ➡
service every minute until a hard state can be determined
    contact_groups admins                ; Notifications ➡
get sent out to everyone in the 'admins' group
    notification_options w,u,c,r         ; Send ➡
notifications about warning, unknown, critical, and recovery events
    notification_interval 60             ; Renotify about ➡
service problems every hour
    notification_period 24x7             ; Notifications ➡
can be sent out at any time
    register       0                     ; DON'T REGISTER ➡
DEFINITION - ITS NOT A REAL SERVICE, JUST A TEMPLATE!
}

```

local-service 服务模板和前面见过的模板一样，只不过多了几个属性。新增属性中的 normal_check_interval 指定 Nagios 检查服务运行是否正常的频率，这里设为 5 分钟。第二个新属性 retry_check_interval 和上一个有关联。如果检查服务时，Nagios 发现服务运行不正常，它会按 max_check_attempts 属性指定的次数进行重试。重试若干次后，若结果不变，则将服务置为运行不正常。在重试期间，会按 retry_check_interval 属性指定的每隔 1 分钟检测一次，而不是按 normal_check_interval 属性指定的每隔 5 分钟检测一次。

■注：Nagios 有软状态和硬状态两种概念。检测失败时，Nagios 还要继续按 max_check_attempts 规定的次数进行检测。达到尝试检测的次数之前，主机或服务都被标识为软失败状态。当检测尝试达到了规定的次数且产生了警告消息后，主机或服务才变成硬失败状态。软失败模式表示，如果一个主机或服务暂时失败，然后又自动恢复，则管理员不会收到警告信息，从而可以减少监控系统中潜在的假性警报数量。更多信息请参考“http://nagios.sourceforge.net/docs/2_0/statetypes.html”。

请注意，此模板还有一个父模板 generic-service，这里不对其做详细讨论。模板中使用到的选项释义可以在以下链接处找到，见 http://nagios.sourceforge.net/docs/2_0/xodtemplate.html#service。

■注：Nagios 试着使监控更为智能，除非运行在主机上的服务运行出了问题，一般不对主机进行检测。但若一个主机上的服务失败了，Nagios 通常也会安排对其底层的主机进行检测。更多信息请参考“http://nagios.sourceforge.net/docs/2_0/checkscheduling.html#host_checks”。

下面再来看另一个服务定义，此定义用来对一个基于网络的服务进行监控，如列表 18-22 所示。

列表 18-22 一个基于网络的服务定义

```
define service{
    use                local-service      ; Name of service template to use
    host_name          gateway.example.com
    service_description Check SMTP
    check_command       check_smtp!25
}
```

在列表 18-22 中，定义了一个新的服务，名为 `check SMTP`，此服务引用了 `local-service` 模板，并定义 `check_command` 为 `check_smtp!25`，并将数值 25 传递给了命令 `check_smtp`。下面来看一下命令的定义。

```
define command{
    command_name       check_smtp
    command_line       $USER1$/check_smtp -H $HOSTADDRESS$ -p $ARG1$
}
```

此命令运行 Nagios 插件 `check_smtp`。它接受 `$HOSTADDRESS$` 宏作为参数，此宏中有要检测的 SMTP 服务器的 IP 地址。`-p` 选项指定通信端口（插件默认为 25），并使用宏 `$ARG1$` 传递端口参数。

可以在 `nagios.log` 日志文件中看到如下所示的服务产生的警报信息。

```
[1235270465] SERVICE ALERT: gateway.example.com;Check SMTP;CRITICAL;HARD;4;➡
CRITICAL - Socket timeout after 10 seconds
```

`nagios.log` 条目包含了 UNIX 纪元的时间（1235270465，也就是 2009 年 2 月 22 日，周日 13: 41: 05）、警报类型、主机和服务以及包含在插件输出中的警报性质。

■注：可以在线转换纪元时间，见 “<http://www.epochconverter.com/>”。

远程监控

到现在为止，只讨论了如何在本机上运行监控服务，比如本地硬盘，或可以通过网络访问的服务，如 SMTP、IMAP 或 SSH 等。Nagios 还可以对暴露在网络上的主机里的服务进行远程监控。Nagios 带有各种不同的命令以监控各种不同的远程主机，其中两个主要的机制是 NRPE 和 `check_by_ssh` 插件（参见 http://www.nagios.org/faqs/viewfaq.php?faq_id=59 的 FAQ 条目）。

NRPE 是一个允许在远程主机上运行 Nagios 插件并将结果返回 nagios 守护进程的工具。可以在 <http://nagios.sourceforge.net/docs/nrpe/NRPE.pdf> 找到 NRPE 有关的文档。

`check_by_ssh` 插件允许通过 SSH 登录到一个远程主机并执行命令，返回结果。此插件已经在本章中介绍过。

更多监控方式的信息，请参考如下文档。

使用 SNMP 监控：http://nagios.sourceforge.net/docs/3_0/monitoring-routers.html。

监控 Linux/Unix: http://nagios.sourceforge.net/docs/3_0/monitoring-linux.html。

监控 Windows: http://nagios.sourceforge.net/docs/3_0/monitoring-windows.html。

简单的远程监控

可以使用 Nagios 插件创建各种不同的基于网络的服务,但是如果要监控的服务并不面向网络或不在本地主机上时,应当如何呢?对这个问题的其中一个解决方法是使用一个特殊的插件 `check_by_ssh` (其他方式请参考“远程监控”的介绍)。

`check_by_ssh` 插件使用 SSH 连接到一个远程主机并执行命令。因此,要使用此插件,远程主机上必须运行一个 SSH 守护进程,同时任何干预防火墙必须允许 SSH 信息进出此主机。

除此之外,还需要在主机之间使用基于密钥的认证机制,因为 Nagios 对服务进行检测时没有输入密码的功能。下面就从在 Nagios 服务器和远程主机之间创建一个密钥开始。

■注: 在第 9 章中对基于密钥的 SSH 认证机制做过介绍。

创建此密钥之前,需要先使用运行 Nagios 的用户(通常是 `nagios`)登录到主机。可以使用 `su` 命令进行此操作。

```
$ sudo su - nagios
```

然后使用 `ssh-keygen` 命令创建一个密钥。

```
nagios$ ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/var/log/nagios/.ssh/id_dsa):
Created directory '/var/log/nagios/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /var/log/nagios/.ssh/id_dsa.
Your public key has been saved in /var/log/nagios/.ssh/id_dsa.pub.
The key fingerprint is:
41:32:dd:36:e4:cc:2a:7e:5e:01:10:00:b6:99:5f:b1 nagios@au-mel-rhel-1.example.com
```

这里使用“`-t dsa`”选项来创建一个 DSA 密钥。命令提示输入密钥的存放位置,此位置通常是 `nagios` 用户 `home` 目录下的 `.ssh` 目录,此例中是 `/var/log/nagios/.ssh`。私有密钥在 `id_dsa` 文件中,公有密钥在 `id_dsa.pub` 文件中。由于希望建立的连接没有密码或密码提示,因此在这里没有输入密码字符,而是直接按回车键指定一个空密码。

然后,需要把公共密钥 `id_dsa.pub` 复制到远程主机上并为要连接到的用户将其保存在 `authorized_key` 文件中。如果读者一直跟随这个实例的操作,现在应该在远程主机上创建一个用户并为其指定一个密码。示例中,在远程主机 `gateway.example.com` 中进行如下操作。

```
gateway$ sudo useradd nagios
gateway$ sudo passwd nagios
```

还要在远程主机上创建 `.ssh` 目录并建立保护。

```
gateway$ mkdir /home/nagios/.ssh
gateway$ chmod 0700 /home/nagios/.ssh
```


下面就可以复制文件了。

```
nagios$ scp .ssh/id_dsa.pub nagios@gateway.example.com:~/.ssh/authorized_keys
```

如果复制成功，就可以从 Nagios 服务器使用 SSH 登录 gateway 主机，登录的过程中并不要求输入密码。

```
nagios$ ssh nagios@gateway.example.com
```

下面配置 Nagios 来使用此连接检测服务。`check_by_ssh` 插件也要求运行的命令安装在远程主机上。此命令通常是一个 Nagios 插件，因此最简单的方式就是在远程主机上安装 Nagios 插件软件包。在 Red Hat 中，运行如下命令。

```
gateway$ sudo yum install nagios-plugins
```

在 Ubuntu 中的命令如下所示。

```
gateway$ sudo apt-get install nagios-plugins
```

接下来要定义一个使用 `check_by_ssh` 插件监控远程主机服务的命令。例如，要监控远程主机的负载情况，可以使用如下命令。

```
define command {
    command_name    check_load_ssh
    command_line    $USER1$/check_by_ssh -H $HOSTADDRESS$ -l nagios -C "/usr/lib/nagios/plugins/check_load -w $ARG1$ -c $ARG2$"
}
```

命令名为 `check_load_ssh`。`command_line` 指定运行 `check_by_ssh` 插件并使用 `-H` 选项连接到主机。

`-l` 选项指定要连接到的远程主机上的用户名，这里使用了刚刚创建的 `nagios` 用户。

`-C` 指定要在远程主机上运行的命令。此例中，运行另外一个本地安装的名为 `check_load` Nagios 的插件，并把 `-w` 和 `-c`（WARNING 和 CRITICAL）线程的值作为参数传递给了此命令。

■提示： `check_by_ssh` 命令可以完成很多工作。使用 `--help` 选项运行此命令可以看到其所有的功能。

一旦定义好了命令，下面就可以定义要使用此命令的服务了，如下所示。

```
define service {
    use                local-service                ; check current
load on machine
    service_description    Current Load
    check_command          check_ssh_load!5.0,4.0,3.0!10.0,6.0,4.0
}
```

服务名为 `Current Load`，运行 `check_ssh_load` 命令并传递了两个参数，参数指定平均负载超过 1 分钟、5 分钟和 15 分钟时，触发 WARNING 和 CRITICAL 状态。

■注： 在第 17 章中讨论过负载的相关话题。

以上展示的是如何进行远程监控的一个简单的示例，此示例并不理想（基于密钥的 SSH 连接可能会造成安全漏洞），但它却提供了一个最快和最简单的方法。对于比较复杂的环境，NRPE 服务器以及相应的命令通常应该是一个更好的选择。

■注：就像在主机对象的定义时一样，这里的讨论也没有包含服务相关的所有可用功能。也可以将服务进行分组，使其相互依赖，以及在这些服务上运用其他各种有用的技巧等。建议阅读可用的 Nagios 文档和许多其他有用的资源并从中获取更多信息。

4. Nagios 插件

有大量可用的插件可以用来创建服务以及对服务进行检查的命令。表 18-11 列出了部分可用的插件。

表 18-11	Nagios 插件
插件	描述
check_ntp	检查 NTP 服务
check_swap	检查交换空间
check_ifstatus	检查网络接口
check_tcp	检查基于 TCP 的网络服务
check_by_ssh	使用 SSH 检查服务
check_imap	检查一个 IMAP 服务
check_clamd	检查 ClamAV 守护进程
check_udp	检查基于 UDP 网络服务的状态
check_dig	使用 dig 检查 DNS 服务
check_ping	通过 ICMP 检查主机状态
check_simap	检查一个 IMAP 服务
check_nagios	检查 Nagios 进程状态
check_snmp	通过 SNMP 检查
check_http	检查 Web 服务器的状态
check_ssh	检查 sshd 服务的状态

表中列出的只是所有可用插件中的一小部分，但这些应该可以为创造一个适合环境的检查打下一个良好的基础了。多数 Nagios 插件都非常简单并易于理解。而且，其中大多数都提供--help 选项来显示其功能和选项。

还有很多 Nagios 插件软件包之外的插件。比如，可以在如下链接处找到很多这样的插件，见“<http://www.nagiosexchange.org/cgi-bin/page.cgi?d=1>。”

■注：可以在以下链接处找到一些有用的附加信息，见“<http://www.nagios.org/download/addons/>。”

如果需要，还可以自己开发插件。如下链接处有自己开发插件的有关指南和实例，见“http://nagios.sourceforge.net/docs/2_0/plugins.html#howto。”

18.3.5 搭建 Nagios 控制台

到现在为止，读者应该已经明白了主机和服务是如何被定义的，读者还可以创建自己的主机和服务来作为两个发行版本中所提供的示例的补充。一旦创建了主机和服务后，有一个控制台对其进行观察将是非常有用的。Nagios 自带一个功能完整的基于 Web 的控制台，下面将来讨论如何搭建它。

■提示：也存在 Nagios 控制台的替代品。可以在如下链接处找到这些替代品的列表，见“<http://www.nagiosexchange.org/cgi-bin/page.cgi?g=Utilities%2FGUI%2Findex.html;d=1>。”

Nagios 控制台可以运行在第 11 章中介绍的 Apache 服务器中，Red Hat 和 Ubuntu 中都默认安装了此 Web 控制台。在 Red Hat 上，Nagios 的 Apache 配置文件在/etc/httpd/conf.d/nagios.conf；在 Ubuntu 上，是/etc/apache2/conf.d/nagios2.conf。

Red Hat 中，控制台位于 Web 服务目录/nagios；Ubuntu 中，位于/nagios2。如果 Web 服务器正在运行，则分别在 Red Hat 和 Ubuntu 的浏览器中输入“http://headoffice.example.com/nagios/”和“http://headoffice.example.com/nagios2/”（将主机名称换成读者自己 Nagios 主机的名称）就能显示出控制台页面。

■注：可以使用在第 11 章中所学的内容将 Nagios 控制台移动到一个虚拟主机或其他位置上。

1. 控制台认证

为了防止有人恶意使用 Nagios 控制台，Web 服务器设置了一些基本的身份认证机制。Nagios 使用 Apache 基本的 HTTP 认证来保护控制台。当打开控制台时，会弹出输入用户名和密码的提示框，如图 18-2 所示。



图 18-2 Nagios 认证

Apache 通过在服务器上指定一个包含用户名和密码的文件来配置基本的认证，收到基本认证保护的站点会对此文件进行查询。

提醒：这只是最基本的认证。Apache 基本认证使用的密码只经过非常简单的加密，极易被拦截和破解。要想更好的保护 Nagios 控制台，应该考虑为其启用 SSL。

下面来看一下 Apache 的 Nagios 配置文件内部是如何设置密码文件位置的。在 Red Hat 的/etc/httpd/conf.d/nagios.conf 文件中，可以找到如下内容。

```
AuthName "Nagios Access"
AuthType Basic
AuthUserFile /etc/nagios/htpasswd.users
Require valid-user
```

在 Ubuntu 的/etc/apache2/conf.d/nagios2.conf 文件中有如下内容。

```
AuthName "Nagios Access"
AuthType Basic
AuthUserFile /etc/nagios2/htpasswd.users
Require valid-user
```

AuthName 和 AuthType 指令用来启用基本的认证机制。AuthUserFile 指定密码文件地址，Red Hat 中在/etc/nagios/htpaawd.users，Ubuntu 中在/etc/ nagios2/htpaawd.users，这就是要在其中添加用户的两个文件。“Require valid-user” 指令表示只有在此文件中指定的用户可以登录控制台。

知道了认证文件的位置后，就要在相应的位置创建该包含用户名和密码的文件。在 Red Hat 中，使用 htpasswd 命令；在 Ubuntu 中使用 htpasswd2 命令创建此文件。

列表 18-23 列出了命令的执行过程。

列表 18-23 使用 htpasswd 命令

```
$ sudo htpasswd -c /etc/nagios/htpasswd.users jsmith
New password:
Re-type new password:
Adding password for user jsmith
```

htpasswd 命令有两个参数：包含用户名和密码的文件位置以及要添加的用户名。其中还使用了一个命令行开关项——-c。当首次创建密码文件时使用-c 开关项。首次创建完后，就可以不再使用此开关项，只是指定文件和要添加的用户就可以了。

在列表 18-23 中，使用-c 开关项创建了一个名为 htpasswd.users、位于/etc/nagios 下的新的密码文件。同时指定了要添加的用户名 jsmith，然后按命令提示为用户输入密码及确认密码。两次输入的密码必须一致。若密码一致，命令将成功执行，并在指定的文件中添加一个用户。

接下来，就可以使用刚刚添加的用户名和密码登录到 Nagios 控制台并显示控制台窗口了，如图 18-3 所示。

一旦创建了用户名和密码，就可以登录到控制台了，但除此之外还要对用户的访问内容进行配置。可以通过编辑一个名为 cgi.cfg 的文件完成此配置，此文件在 Red Hat 中位于/etc/nagos 目录下，在 Ubuntu 中，位于/etc/nagios2 目录下。

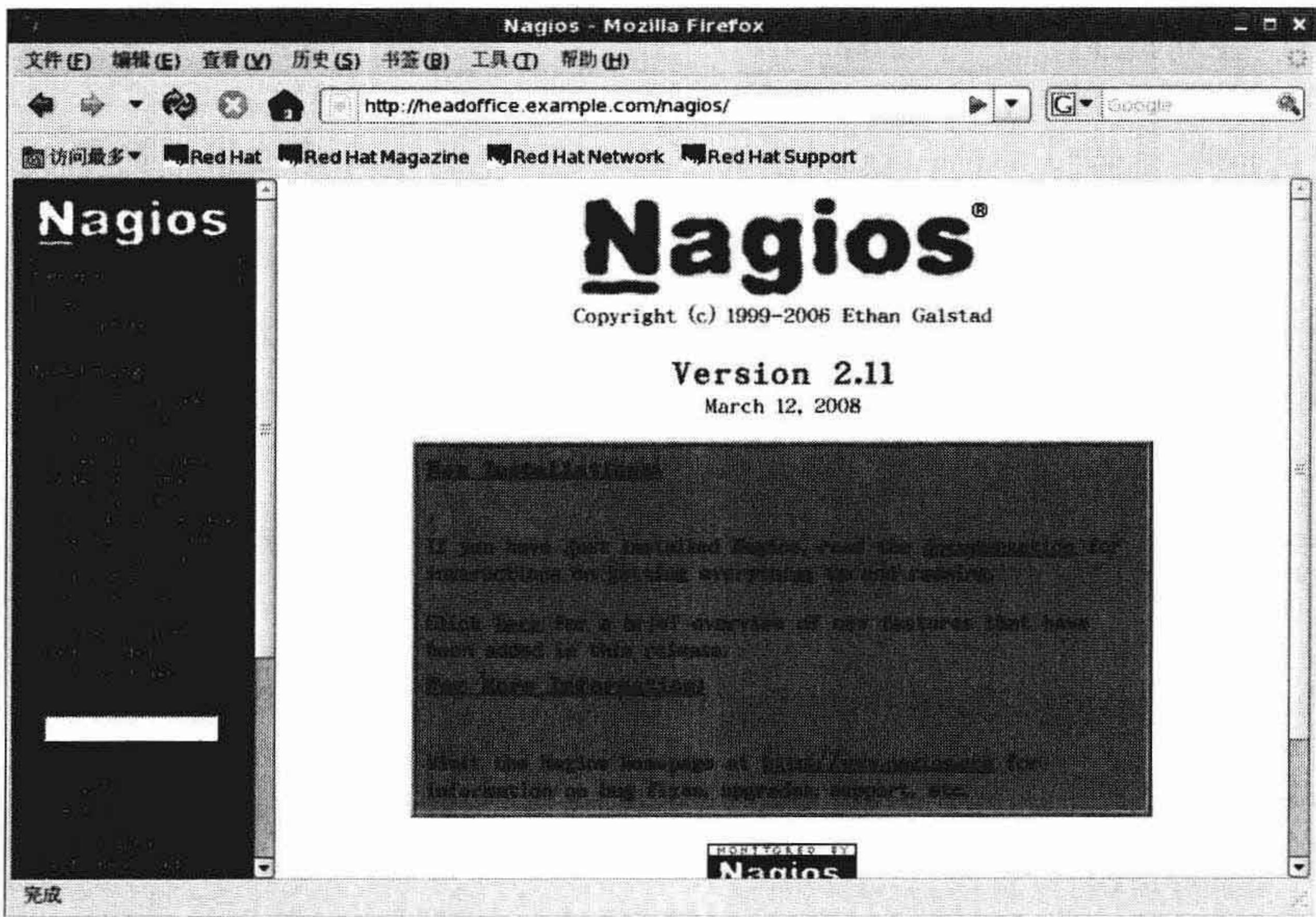


图 18-3 Nagios 控制台

Nagios 控制台中有两种类型的用户：认证用户和认证联系人。两种类型的用户都需要一个用户名和一个密码来登录控制台。认证用户的可访问内容在配置文件 `cgi.cfg` 中指定。认证联系人也是认证用户，只不过每一个认证联系人的用户名必须与一个 Nagios 联系人一致。

因此，比如说，如果用户名 `jsmith` 和一个由 `contact_name` 指令指定的联系人的名字一致，则此认证用户就变成了一个认证联系人。

那么两者有什么不同呢？认证用户被授予了可以查看 Web 控制台的通用权利，而认证联系人被赋予了更多的权利，他们可以查看，也可以操作他们作为联系人的主机和服务。

下面来看一下 `cgi.cfg` 文件。`cgi.cfg` 文件中的第一个指令是 `use_authentication`，此指令用来控制是否为 Nagios 的 Web 控制台启用认证以及 Nagios 是否使用 Web 服务器所提供的认证证书。指令如下所示。

```
use_authentication=1
```

默认设置为 1，表示启用认证，设为 0 时则禁用认证。

在 `cgi.cfg` 文件中还提供了各种指令对控制台的特定功能进行认证，设置时使用用户列表作为选项，各个用户之间用逗号隔开。例如：

```
authorized_for_all_services=jsmith,nagiosadmin
```

`authorized_for_all_services` 指令控制可以在控制台上查看服务的用户，这里指定 `jsmith` 和 `ngiosadmin` 拥有此访问权限。

表 18-12 中是所有可用的认证命令及其解释。

表 18-12 认证指令

指令	解释
authorized_for_system_information	可以访问 Nagios 进程信息的用户
authorized_for_configuration_information	可以查看所有配置信息的用户
authorized_for_system_commands	可以在控制台中运行命令的用户
authorized_for_all_services	可以访问所有服务的用户
authorized_for_all_hosts	可以访问所有主机的用户
authorized_for_all_service_commands	可以运行服务相关的外部命令的用户
authorized_for_all_host_commands	可以运行主机相关的外部命令的用户

表 18-12 中的第一个指令 `authorized_for_system_information` 允许查看 Nagios 进程和服务信息，如进程何时启动以及服务器进行了何种配置等。

第二个指令 `authorized_for_configuration_information` 允许查看监控环境的所有配置信息以及对象定义信息。其中包括主机的配置、服务、联系人、命令以及其他对象类型等。

第三个指令 `authorized_for_system_commands` 对可以在控制台中启动、停止或重启 Nagios 进程的用户进行控制。

接下来的两个指令是 `authorized_for_all_services` 和 `authorized_for_all_hosts`，控制可以查看 Web 控制台中所有主机和服务信息的用户。请记住，认证联系人可以查看其作为联系人的主机和服务的信息。

表 18-12 中的最后的两个指令是 `authorized_for_all_service_commands` 和 `authorized_for_all_host_commands`，分别指定允许哪些用户对主机和服务运行外部命令。授权的用户可以进行一系列操作，如禁用对主机和服务的活动性检查，以及启用或禁用对主机和服务发出警告信息。

■注：在 `cgi.cfg` 文件中，上述所有指令都默认被注释掉了。如果要使用这些指令，需要去掉它们前面的注释符，并在指令中添加所需的用户。

如果要指定所有用户都可以访问某个特定的功能，可以使用*，如下所示。

```
authorized_for_all_services=*
```

此指令设置将允许所有认证用户查看在 Nagios 服务器上定义的所有服务的信息。*对所有的认证指令都适用。

前面提到过，除了赋予他们的权利，作为联系人的用户还可以访问他们作为联系人的主机和服务。对服务来讲，这些访问包括以下几个方面。

- 查看服务状态。
- 查看服务配置。
- 查看服务历史和警告信息的能力。
- 对服务运行命令的能力（比如启动或停止检测等）。

对主机来讲，这些访问包括以下几个方面。

- 查看主机状态。
- 查看主机配置。
- 查看主机历史和警告信息的能力。
- 对主机运行命令的能力（比如启动或停止检测等）。

因为是某个主机的联系人而可以对某个特定主机进行访问的认证联系人也可以对在此主机上的所有服务进行访问，就像他也是这些服务的联系人一样。比如，如果某个用户是 `blah` 主机的认证联系人，则他可以查看此主机上定义的所有服务的状态、配置、服务历史以及警告等信息，还可以对这些服务运行相关的命令。

2. 控制台功能

Nagios 控制台不但提供了一个可以查看主机和服务信息的界面，还提供了对如何监控主机和服务进行控制的能力。下面就来看几个控制台页面，同时建议读者自己浏览控制台上的视图、列表和控制页面。

对于环境状态概要信息，最好的视图是“Tactical Monitoring Overview”页面，如图 18-4 所示。

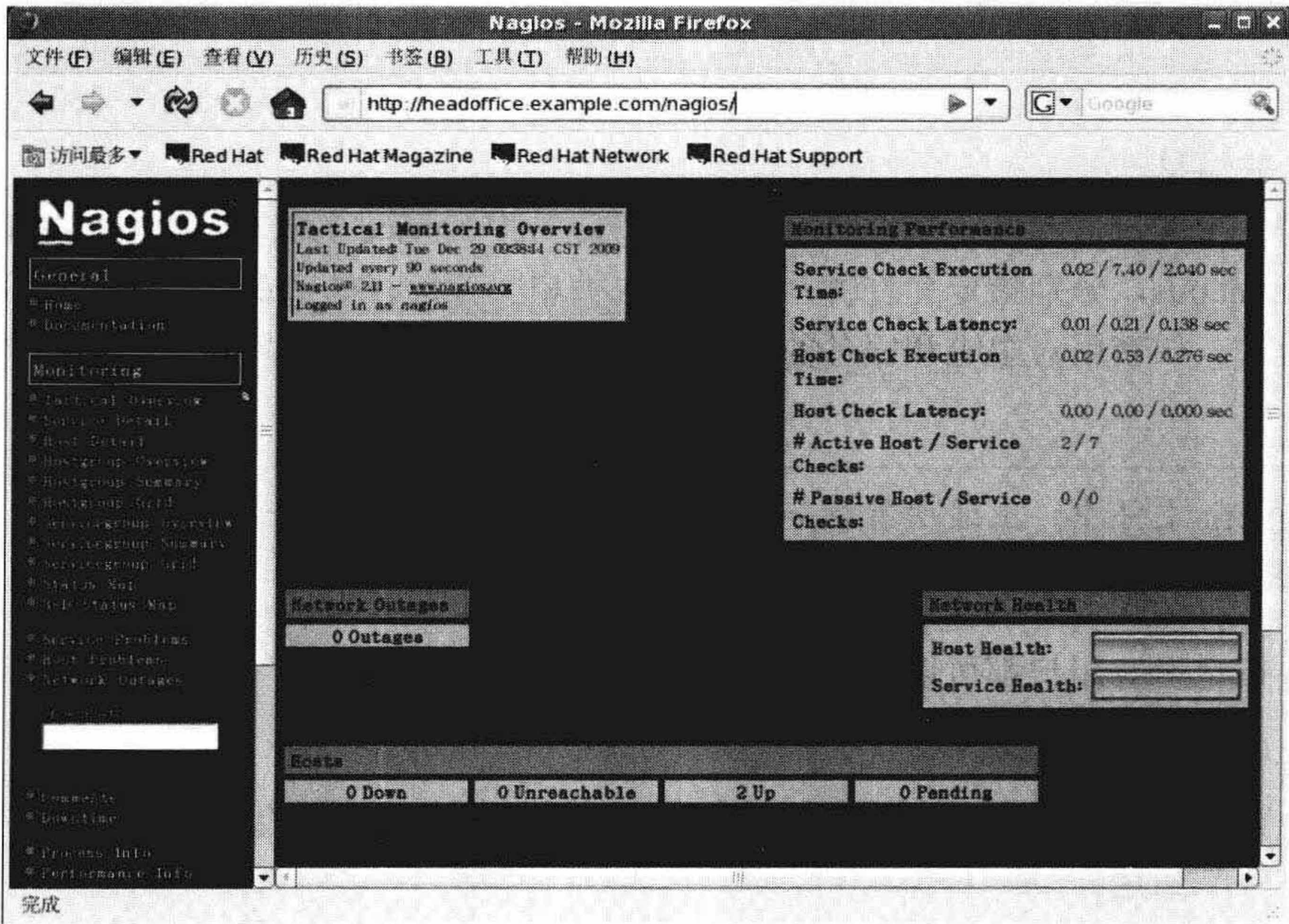


图 18-4 “Tactical Monitoring Overview” 页面

此页面显示了 Nagios 服务器的当前状态及主机和服务器的概要信息。单击左侧菜单中的“Tactical Monitoring Overview”链接就可以打开此页。

要查看主机和服务的更多细节，左侧菜单中的“Host Detail”和“Service Detail”链接可以显示所监控主机和服务的完整列表。图 18-5 所示的是“Service Detail”页面。

图 18-6 所示的是“Host Detail”页面。

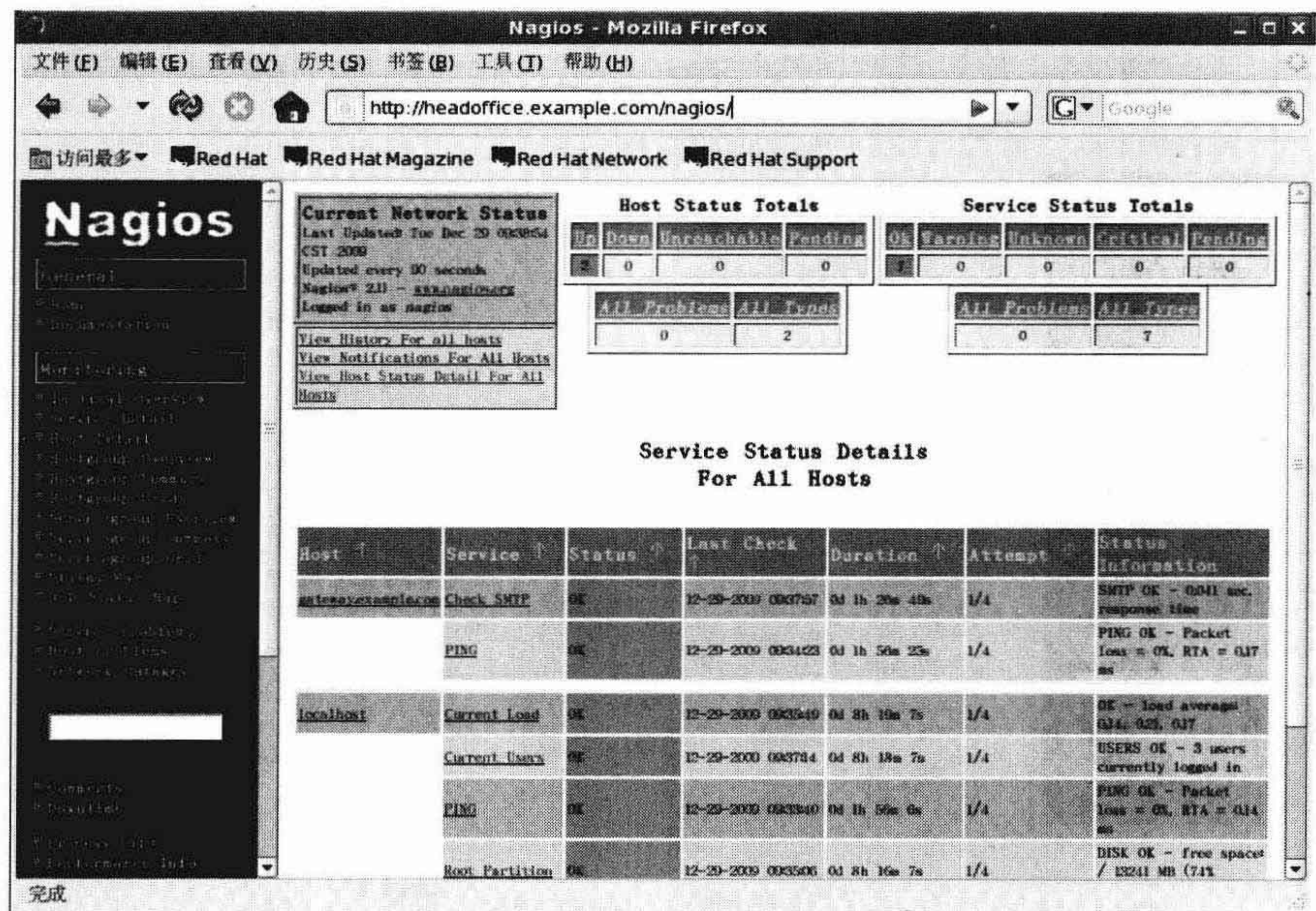


图 18-5 “Service Detail” 页面

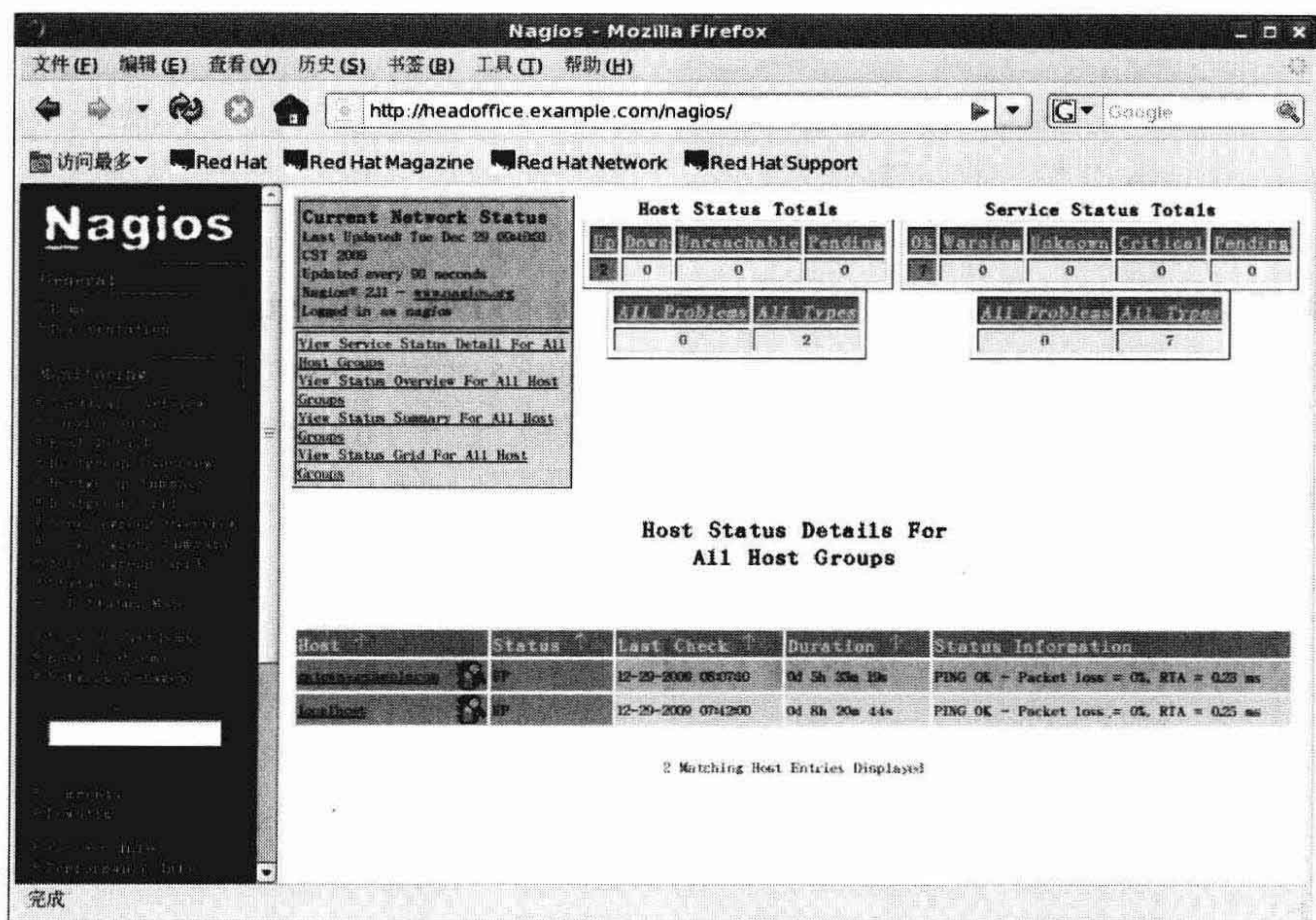


图 18-6 “Host Detail” 页面

在两个页面上，都可以单击主机或服务名来展开其状态和配置信息。

最后一个控制台页面是如图 18-7 所示的“Process Info”页面，此页面显示了 Nagios 进程的状态和信息。可以使用此页面对 Nagios 的选项进行多种配置，比如启用或禁用检查和警告等。

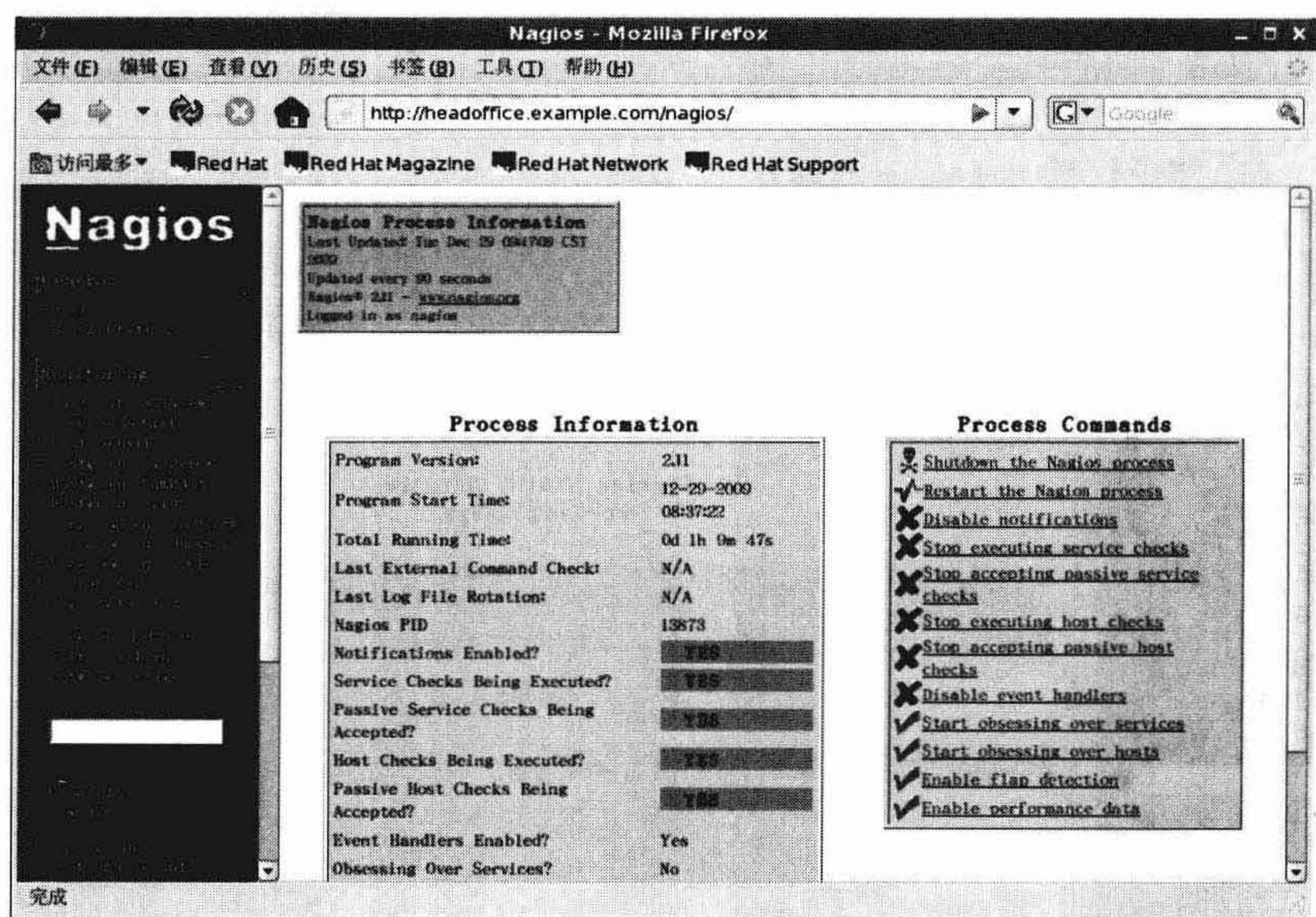


图 18-7 Nagios “Process Info” 页面

18.3.6 Nagios 疑难解答

有很多可用资源（包括网站、论坛以及书籍）都可以帮助读者使用 Nagios，比如本书的一位作者 James Turnbull 编写的《精通 Nagios 2.0》（Apress，2006）。

还有很多的可供选择的支持，包括商业支持见“<http://www.nagios.org/support/>”，可以理解的文档，见“<http://www.nagios.org/docs/>”，以及 Nagios 的 FAQ，见“<http://www.nagios.org/faqs/>”。可以求助于活跃的有帮助的邮件列表“<http://www.nagios.org/support/maillinglists.php>”，还可以在 Nagios 论坛和 wiki 中查找答案，两者都位于“http://community.nagios.org/wiki/index.php/Main_Page”。

18.4 小 结

本章中，学习了日志和监控以及如何让它们开始工作。讨论了 syslog 是如何工作的以及如何最大限度地利用 syslog 守护进程。还讨论了如何使用 SEC 搜索日志数据以找到对管理环境较重要的日志条目。

另外，本章还探讨了 Nagios（一个企业监测工具）。看到了如何安装和配置 Nagios，也看到了如何配置主机和服务，以及如何为监测环境设置支持的配置。

下一章将讨论主机配置，以及如何使用自动化配置工具来管理配置信息。

第 19 章 配置管理

James Turnbull 编写

本章中将会介绍两个方面的配置管理，如下所示。

- 自动配置并安装新主机。
- 自动配置管理，包括文件、用户和软件包等。

介绍的第一个过程是自动配置并安装主机，有时也称为 bootstrapping。在 Red Hat 中，bootstrapping 也通常被称为 kickstarting（使用 Kickstart 工具来运行）。在 Ubuntu 和 Debian 中，此过程被称为 preseeding。

自动配置是在主机中自动安装发行版的一个方法。当在第 2 章中首次介绍发行版的安装时，介绍了如何手动安装。插入一个 DVD 并按照屏幕上的提示就可以安装发行版了。自动配置是无需按照提示输入配置问题而自动安装发行版的一种方法。这使得配置变得很简单和迅速，同时其优点还在于能使每一个安装都是相同的。

■提示：可以在服务器主机上或桌面主机上使用配置。它不仅是安装（或重新安装）服务器主机的一个快速方法，还是为用户快速自动安装桌面主机的快速方法。

介绍的第二个过程是配置管理与自动化技术。现在读者已经看到了，用户可以累积安装很多软件包、用户配置文件和其他配置。因此若不采取控制措施和使用自动化处理，用户环境可能会迅速变得复杂和难以管理。配置管理允许用户将配置集中、记录并自动执行。这就使得用户可以管理和控制环境变化，并能保护配置免受意外或恶意更改。

若读者部署了许多主机，则自动配置和配置管理都将是非常有用的，当然对于小型环境它们也都是适用的，可以在管理主机时节省时间和精力。

19.1 自动配置

前面已经简单了解了什么是自动配置，但其在各发行版中的使用却不尽相同。下面将介绍如何在 Red Hat 和 Ubuntu 主机中进行自动化配置。

自动配置通常分为两个阶段进行，如下所示。

1. 启动主机并准备好安装所需的文件。
2. 自动完成各个安装步骤。

此过程开始于启动主机。还记得在第 5 章学过的启动流程吗？在很多主机中，都可以查找替换位置获取启动指令来配置启动顺序，比如从 DVD 或 U 盘启动。除了这些，还可以从网络启动。

在启动进程结束之后的称为预启动运行环境 (PXE)，因此一个网络启动的服务器就被称为 PXE（读作 *pixie*）启动服务器。要安装的主机使用网络查询来发现一个 PXE 启动服务器，网络通常会查询到一个 DHCP 服务器，此服务器会提供主机启动所需的文件并使用一个称为普通文件传输协议 (TFTP) 的文件传输协议将文件传送给主机。

■注：关于 PXE 的更多信息，参见http://en.wikipedia.org/wiki/Preboot_Execution_Environment。

一旦启动初始化完成后，自动配置过程会继续安装一个预包装版本的发行版本，通常安装的过程中会由脚本自动回答安装程序提出的各种配置问题。

■注：我们使用基于网络的自动配置方式来创建一个主机而不是其他替代方式，如 CD 或 DVD 等。因为我们相信基于网络的自动配置方式是自动安装主机最简单、最易用和最有效的方式。

本章中，将会介绍一些有用的工具。Red Hat 中的自动配置使用 Cobbler 工具，这是一款自动化安装框架。Cobbler 也是调用 Red Hat 的安装自动化工具 Kickstart。在 Ubuntu 中，也是介绍如何配置网络启动服务器，同时也介绍如何使用 Kickstart（还要补充一些 Ubuntu 预配置工具的内容）来安装主机。

19.1.1 在 Red Hat 中使用 Cobbler 自动化配置

Red Hat 有几种用于自动配置主机的工具，从最基本的自动化安装工具 Kickstart，到诸如 Cobbler (<https://fedorahosted.org/cobbler/>)、Spacewalk (<http://www.redhat.com/spacewalk/>) 以及 Genome (<http://genome.et.redhat.com/>) 的全功能 GUI 主机配置管理工具。

这里将介绍结合使用如下两个工具。

- Kickstart：一款安装自动化工具。
- Cobbler：一款提供 PXE 启动服务的自动配置服务器。

接下来将介绍创建一个 Cobbler 服务器和搭建安装的过程。在本章的后续内容中，还将介绍如何配置 Kickstart 来进行自动配置和设定安装选项。

1. 安装 Cobbler

现在开始在主机上安装 Cobbler。要运行 Cobbler，需要先安装一些先决条件软件包。

```
$ sudo yum install yum-utils createrepo dhcp tftp-server httpd
```

这里，安装了一些附加的 Yum 工具及 Createrepo 软件包，这是一个库文件管理协助软件。还安装了一些 Cobbler 要用到的附加软件包：DHCP 守护进程，TFTP 服务器和 Apache Web

服务器。若用户已经安装了这些软件包，Yum 将会自动跳过。

■注：DHCP 在第 9 章讨论过，Apache 在第 11 章讨论过。

接下来要安装的是最新版本的 Cobbler 和另一个必须的软件包 `python-cheetah`，此软件包用于协助配置 Kickstart。可以从 Fedora EPEL 库中下载并安装这些软件包。

```
$ sudo rpm -Uvh http://download.fedora.redhat.com/pub/epel/5/i386/cobbler-1.4.1-1.el5.noarch.rpm http://download.fedora.redhat.com/pub/epel/5/i386/python-cheetah-2.0.1-1.el5.i386.rpm
```

或者可以先把 EPEL 库添加到主机中，然后从库中安装这些软件包。如果没有在 Yum 配置中添加 EPEL 库，可以通过安装 `epel-release` RPM 包来添加。

```
$ sudo rpm -Uvh http://download.fedora.redhat.com/pub/epel/5/i386/epel-release-5-3.noarch.rpm
```

下面就可以安装 `cobbler` 软件包了。

```
$ sudo yum install cobbler
```

2. 配置 Cobbler

安装完所需的软件包后，需要对 Cobbler 进行配置。Cobbler 自带了一个非常方便的检查功能，可以告诉用户如何对其进行配置。要了解应进行哪些配置，运行如下命令。

```
$ sudo cobbler check
The following potential problems were detected:
#0: The 'server' field in /etc/cobbler/settings must be set to something other than localhost, or kickstarting features will not work. This should be a resolvable hostname or IP for the boot server as reachable by all machines that will use it.
#1: For PXE to be functional, the 'next_server' field in /etc/cobbler/settings must be set to something other than 127.0.0.1, and should match the IP of the boot server on the PXE network.
#2: service cobblerd is not running
#3: change 'disable' to 'no' in /etc/xinetd.d/tftp
#4: since iptables may be running, ensure 69, 80, 25150, and 25151 are unblocked
#5: fencing tools were not found, and are required to use the (optional) power management features. install cman to use them
```

可以看到要运行 Cobbler 需进行几项设置。下面将逐个解决。

首先，配置文件 `/etc/cobbler/settings`。需要升级此文件中的两个字段，即 `server` 和 `next_server`。还需要使用主机的 IP 地址替换原有的值（通常是 127.0.0.1），这样基于 PXE 启动的主机就可以找到用户的 Cobbler 主机了。此例中，指定如下所示的字段。

```
server 192.168.0.1
next_server 192.168.0.0.1
```

运行如下命令升级 Cobbler 配置。

```
$ sudo cobbler sync
```

■注：任何时候修改了文件 `/etc/cobbler/settings` 后都要运行 “`$sudo cobbler sync`” 命令对配置进行更新。`settings` 配置文件中的常见错误为选项后遗留了多余的空格。请确保从文件中行删除了多余的空格。

还需要配置一个 DHCP 服务器（就像在第 9 章中介绍过的一样）。有两个选择：可以使用 Cobbler 管理现存的 DHCP 服务器，或者可以告诉现存的 DHCP 服务器指向 Cobbler。

3. 使用 Cobbler 管理 DHCP

如果要启用 Cobbler 来管理 DHCP 服务器，就需要启用/etc/cobbler/settings 文件中的另外一项，如下所示。

manage_dhcp: 1

还需要更新 Cobbler 来管理 DHCP 服务器的一个模板文件，即/etc/cobbler/dhcp.template。列表 19-1 显示的是此文件的一个示例。

列表 19-1 /etc/cobbler/dhcp.template 文件

```
# *****
# Cobbler managed dhcpd.conf file
#
# generated from cobbler dhcp.conf template ($date)
# Do NOT make changes to /etc/dhcpd.conf. Instead, make your changes
# in /etc/cobbler/dhcp.template, as /etc/dhcpd.conf will be
# overwritten.
#
# *****

allow booting;
allow bootp;

ddns-update-style interim;
ddns-ttl 3600;
default-lease-time 600;
max-lease-time 7200;
log-facility local7;

ignore client-updates;
set vendorclass = option vendor-class-identifier;

key dynamic-update-key {
    algorithm hmac-md5;
    secret "3PDRnypPtzJqpbQvbw/B7bhPuHqpUe0Sdi95Z4Ez/IzhS6ldzcK6MJ6CdFHkkegp
TNlkmXOM6GggRNE24aPmOw==";
}

zone 0.168.192.in-addr.arpa. {
    key dynamic-update-key;
    primary 192.168.0.1;
}

zone example.com. {
    key dynamic-update-key;
    primary 192.168.0.1;
}

subnet 192.168.0.0 netmask 255.255.255.0 {
    option routers 192.168.0.254;
    option domain-name "example.com";
    option domain-name-servers 192.168.0.1;
    option broadcast-address 192.168.0.255;
next-server $next_server;
filename "/pxelinux.0";
group "static" {
    use-host-decl-names on;
```



```

        host au-mel-rhel-1 {
            hardware ethernet 00:16:3E:15:3C:C2;
            fixed-address au-mel-rhel-1.example.com;
        }
    }
    pool {
        range 192.168.0.101 192.168.0.150;
        deny unknown clients;
    }
    pool {
        range 192.168.0.151 192.168.0.200;
        allow unknown clients;
        default-lease-time 7200;
        max-lease-time 21600;
    }
}

```

如果有一个已经配置好了的 DHCP 服务器，应该根据其配置更新此文件。可以看出列表 19-1 中的模板已经根据第 9 章中所做的 DHCP 配置进行了更新。在其中添加了如下两项设置。

```

allow booting;
allow bootp;

```

这两项告诉 DHCP 服务器对请求网络启动的主机请求进行响应。

列表 19-1 中的其他两项重要的设置分别是 `next-server` 和 `filename` 配置选项。`next-server` 选项设置为 `$next_server`，此值将会被刚刚在 `/etc/cobbler/settings` 文件中配置给 `next_server` 选项的 IP 值所代替。该值将会告诉 DHCP 服务器把请求网络启动的主机路由至何处。

`filename` 项设置为 `/pxelinux.0`，这是基于 PXE 启动的主机所寻找的用来开始其启动进程的启动文件名称。稍后将会设置此文件。

现在，更改了这些文件后，需要运行如下命令。

```
$ sudo cobbler sync
```

警告：如果有一个 DHCP 服务器，此模板将会通过覆盖配置文件 `/etc/dhcpd.conf` 来覆盖其配置。这样做之前最好清楚自己在做什么，并在运行命令前对文件 `/etc/dhcpd.conf` 做一个备份。

4. 不使用 Cobbler 管理 DHCP

如果不想使用 Cobbler 管理 DHCP，只需要简单地修改现存的 DHCP 配置文件即可。文件位于 `/etc/dhcpd.conf`，在其中添加 `next-server` 和 `filename` 项。下面来更新在第 9 章中使用此选项创建的部分配置文件，如列表 19-2 所示。

列表 19-2 目前的 `dhcpd.conf` 配置文件

```

allow booting;
allow bootp;

subnet 192.168.0.0 netmask 255.255.255.0 {
    option routers 192.168.0.254;
    option domain-name "example.com";
    option domain-name-servers 192.168.0.1;
    option broadcast-address 192.168.0.255;
}

```



```
filename "/pxelinux.0";
next-server 192.168.0.1;
group "static" {
    use-host-decl-names on;
    host au-mel-rhel-1 {
        hardware ethernet 00:16:3E:15:3C:C2;
        fixed-address au-mel-rhel-1.example.com;
    }
}
pool {
    range 192.168.0.101 192.168.0.150;
    deny unknown clients;
}
pool {
    range 192.168.0.151 192.168.0.200;
    allow unknown clients;
    default-lease-time 7200;
    max-lease-time 21600;
}
```

可以看到在 **DHCP** 部分的开始添加了如下的两个选项。

```
allow booting;
allow bootp;
```

这两个选项告诉 **DHCP** 响应来自启动客户端的请求。

另外还在子网定义中添加了 **next-server** 选项。

```
next-server 192.168.0.1
```

next-server 选项告诉 **DHCP** 将 **PXE** 网络启动请求转向何处。这里需要指定 **Cobbler** 服务器的 IP 地址。

最后，添加了 **filename** 选项，设置为 **/pxelinux.0**，这是基于 **PXE** 启动的主机所寻找的用来开始其启动进程的启动文件名称。稍后将会设置此文件。

■提示：配置完 **DHCP** 服务器后，需要重启 **Cobbler** 服务器以应用新的配置。

5. 启动 Cobbler 和 Apache

接下来，启动 **Cobbler** 守护进程。

```
$ sudo service cobblerd start
```

还要确保已经启动了 **Apache Web** 服务器。

```
$ sudo service httpd start
```

6. 配置 TFTP

守护进程启动后，还要确保 **TFTP** 服务器将启动文件发送到要安装的主机中。如此，需要编辑文件 **/etc/xinet.d/tftp** 来启用 **TFTP** 服务器。在其中包含如下行：

```
disable = yes
```

改为：

```
disable = no
```


接下来启用 TFTP 服务器。

```
$ sudo chkconfig tftp on
```

■注：无需启动服务，因为此服务运行在正在运行中的 xinetd 服务之下。若运行“chkconfig - list”命令，将会看到在列表的尾部有 xinetd 服务，也可以看到 TFTP 服务已经启用。

通过创建如下 iptables 规则，打开防火墙上的一些必要端口，比如 69、80、25150 等，确保主机可以连接到 Cobbler 服务器。

```
-A Firewall-eth0-INPUT -s 192.168.0.0/24 -p udp -m state --state NEW --dport 69 -j ➡
ACCEPT
-A Firewall-eth0-INPUT -p tcp -m state --state NEW --dport 80 -j ACCEPT
-A Firewall-eth0-INPUT -s 192.168.0.0/24 -p tcp -m state --state NEW --dport 25150 ➡
-j ACCEPT
-A Firewall-eth0-INPUT -s 192.168.0.0/24 -p tcp -m state --state NEW --dport 25151 ➡
-j ACCEPT
```

这些规则允许 192.168.0.0/24 子网中的任何主机在适当的端口访问启动服务器。更多防火墙规则可以参见第 6 章。

7. 使用 Cobbler

配置完 Cobbler 后，就可以启动并使用它了。Cobbler 允许为安装主机指定一个发行版，引入发行版文件，然后创建一个配置文件，然后用户就可以使用此发行版和配置来安装主机了。

下面使用 import 命令来创建第一个配置文件。

```
$ sudo cobbler import --mirror=/media/cdrom --name=RHEL5 --arch=i386
```

命令中给 Cobbler 命令加上了 import 选项。--mirror 选项指定了要安装的发行版源文件，在这里，使用 DVD 驱动器/media/cdrom，并假定发行版的 CD 或 DVD 挂载在了此驱动器中。

也可以指定一个在线的库文件，如下所示。

```
$ sudo cobbler import --mirror=rsync://ftp.iinet.net.au/pub/fedora/linux/releases/➡
10/Fedora/i386/ --name=Fedora10 --arch=i386
```

此处使用 rsync（一种简单文件传输方式）指定了一个可用的 Fedora 10 安装。Cobbler 会下载所需文件并为 Fedora 安装创建一个发行版本和配置文件。

■提示：需要有足够的磁盘空间来保存要安装的发行版本所有文件。比如，RHEL5 的安装需要 3GB 的空闲空间。

Cobbler 将运行导入进程，然后返回到提示页面。依据主机的性能（如果主机连入了网络，还依赖于网络的速度），可能需要一些时间。

import 命令中最后的两个选项--name 和--arch，分别指定了要创建的配置文件名（比如 RHEL5）和导入的发行版架构（比如 i386、x86_64 等）。Cobbler 通常可以自动检测，但设置后会更加安全。架构类型将会作为后缀添加到创建的配置文件名后（比如 RHEL5-i386）。

创建发行版和配置文件后，可以使用 report 选项在 Cobbler 命令中看到这些设置，如列表 19-3 所示。

列表 19-3 一个 Cobbler 报告

```
$ sudo cobbler report
distro                : RHEL5-i386
architecture         : i386
breed                 : redhat
created               : Tue Feb 24 21:46:33 2009
comment               : rhel5.2
initrd                : /var/www/cobbler/ks_mirror/RHEL5/images/pxeboot/initrd.img
kernel                : /var/www/cobbler/ks_mirror/RHEL5/images/pxeboot/vmlinuz
kernel options        : {}
ks metadata           : {'tree': 'http://@@http_server@@/cblr/links/RHEL5-i386'}
tree build time       : Thu May 1 09:23:47 2008
modified              : Tue Feb 24 21:52:03 2009
mgmt classes          : []
os version             : rhel5
owners                : ['admin']
post kernel options   : {}
redhat mgmt key        : <<inherit>>
template files        : {}
profile                : RHEL5-i386
distro                 : RHEL5-i386
comment               :
created               : Tue Feb 24 21:46:33 2009
dhcp tag              : default
enable menu            : True
kernel options         : {}
kickstart              : /var/lib/cobbler/kickstarts/sample.ks
ks metadata            : {}
mgmt classes           : []
modified              : Tue Feb 24 21:46:33 2009
name servers           : []
owners                 : ['admin']
post kernel options    : {}
redhat mgmt key         : <<inherit>>
repos                  : []
server                 : <<inherit>>
template_files         : {}
virt bridge            : xenbr0
virt cpus               : 1
virt file size          : 5
virt path               :
virt ram                : 512
virt type              : qemu
```

此选项显示了目前导入 Cobbler 中的所有发行版及其相应的配置文件。

■注：读者将会看到，导入一个发行版后会创建多个发行版和配置文件。比如，导入 RHEL 5 后，将会添加 vanilla RHEL 5 发行版和 Xen（这是在第 20 章中将会讨论到的一种虚拟化技术类型）版本的 RHEL 5。

列表 19-3 显示了创建的 vanilla RHEL 5 发行版和配置文件 RHEL5-i386。列表 19-3 中的很多信息对用户来讲都不是很重要，但是还是要注意一下 kickstart 选项，此项的值为/var/

lib/cobbler/kickstarts/sample.ks。此 Kickstart 文件将会自动安装发行版的每一部分。本章稍后会详细地讨论此文件的内容。

可以使用“cobbler profile”命令编辑配置文件来更改其使用哪一个 Kickstart 文件。还可以使用“cobbler profile list”命令列出所有的配置。

```
$ sudo cobbler profile edit --name=RHEL5-i386 --kickstart=/var/lib/cobbler/kickstarts/custom.ks
```

这里修改 RHEL5-i386 配置以使用/var/lib/cobbler/kickstarts/custom.ks Kickstart 文件。

还可以使用 remove 命令移除一个配置文件，也可以使用 copy 命令复制一个。

```
$ sudo cobbler profile copy --name=RHEL5-i386 --newname=RHEL5-i386-new
$ sudo cobbler profile remove --name=RHEL5-i386-new
```

第一个命令将 THEL5-386 复制为 RHEL5-i386-new，第二个命令将删除 rhel5-i386-new 配置及其文件。

■注：查看 cobbler 命令的 man 页可以看到配置文件中的其他可修改选项。

8. 使用 Cobbler 创建主机

现在已经添加了一个配置和一个发行版，可以启动主机并安装系统了。选中一个要创建的主机（或虚拟机）并重新启动。主机可能会在网络上自动搜索一个启动设备，同时还需要修改 BIOS 设置并调整启动顺序。为了从 Cobbler 启动，需要指定主机先从网络启动。

主机启动时，需要从网络上获取一个 IP 地址，DHCP 服务器将会返回一个应答，如图 19-1 所示。

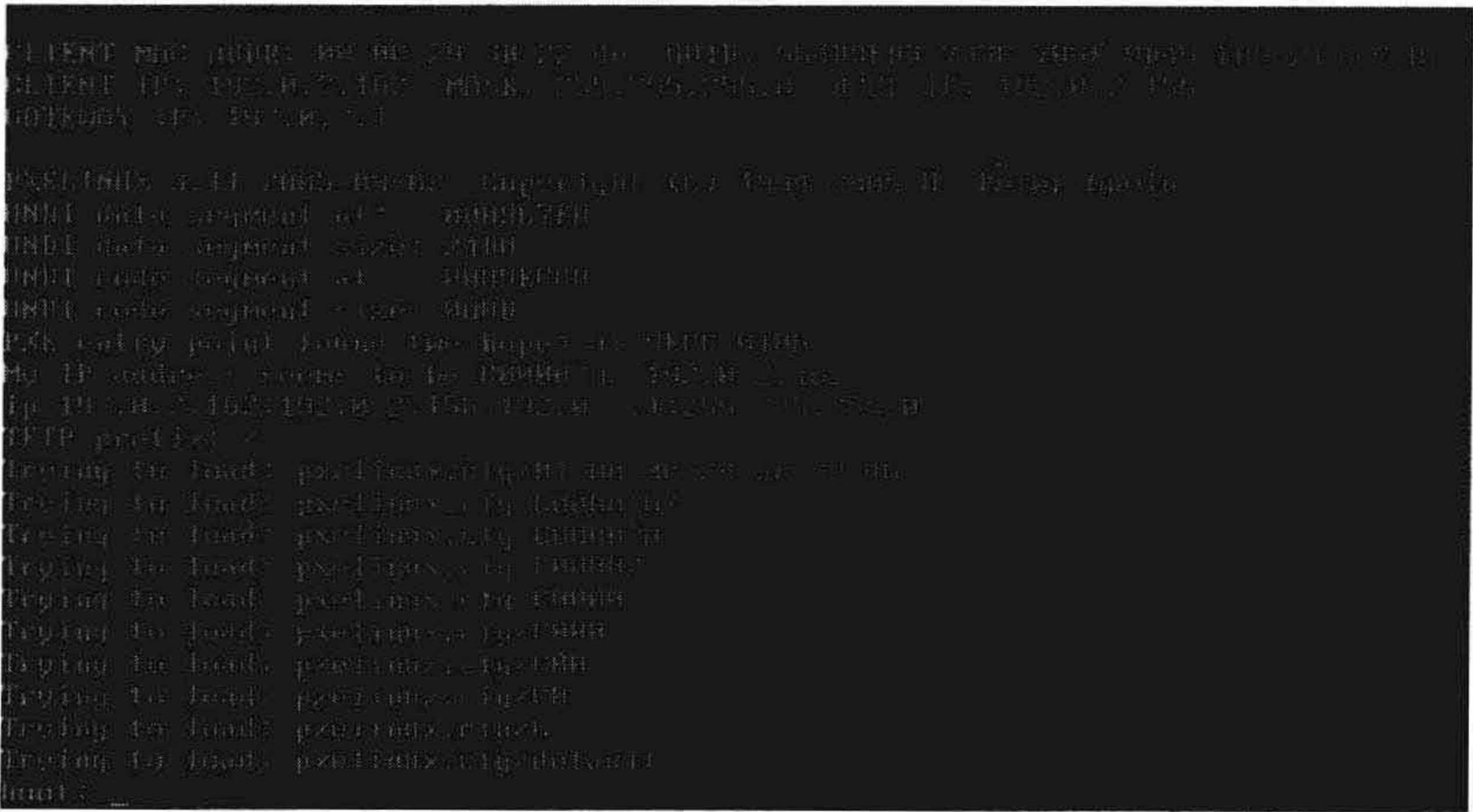


图 19-1 网络启动

此后，主机将会启动到一个称为“boot:”的命令行提示符下。在这儿可以输入 menu 命令来启动 Cobbler 菜单。图 19-2 是此菜单的一个示例。

从这个菜单中，可以选择一个想要安装的配置（比如 RHEL5-i386）。若没有选择要安装的配置，Cobbler 将会自动启动菜单中的第一项，这会在本机中继续启动进程。

9. Cobbler 的 Web 界面

Cobbler 还有一个可以用来对部分选项进行管理的、简单的 Web 界面。此界面的使用很简单，但命令行界面功能更加齐全，如果要使用，也可以部署实现它。可以在网址“<https://fedorahosted.org/cobbler/wiki/CobblerWebInterface>”处找到可用的命令。

10. Cobbler 排错

可以通过监控包括日志文件在内的主机文件来对网络启动过程进行错误排查，还可以使用如 `tcpdump` 或 `tshark` 命令等网络监控工具进行排错。

可以通过查看日志文件 `/var/log/messages` 监控 DHCP 进程的输出信息。Cobbler 还将日志文件记录到文件 `/var/log/cobbler/cobbler.log` 中，同时，包含 `kickstep` 和 `syslog` 目录的文件也在路径 `/var/log/cobbler` 下。

还可以监控在主机和启动服务器间的网络流量。可以使用各种网络监控工具完成此工作。

```
$ sudo tcpdump port tftp
```

在网址“<https://fedorahosted.org/cobbler/wiki/UserDocs>”处有一个包含 Cobbler 文档的 wiki 页面。页面“<https://fedorahosted.org/cobbler/wiki/UserDocs#Troubleshooting>”处有关于错误排查提示的文档。Cobbler 社区在“<https://fedorahosted.org/mailman/ustinfoo/cobbler>”处有一个邮件列表，以及一个在 Freenode 上 `#Cobbler` 处的活跃的 IRC 频道。

19.1.2 在 Ubuntu 中进行自动化配置

就像在 Red Hat 中一样，Ubuntu 也可以自动部署主机，但它没有像 Cobbler 那样的服务器解决方案。为了取得和在 Red Hat 中一样的效果，需要在 Ubuntu 主机上做一些额外的手动配置来进行网络启动。

■注：Ubuntu 可以使用一个名为 Preseed 的工具来自动化安装过程。而且，Ubuntu 也支持 Kickstart。下一节将介绍如何结合 Kickstart 和 Preseed 来配置自动化安装过程。

1. Ubuntu 中的 Cobbler

Ubuntu 也支持 Cobbler，然而，目前却没有为 Ubuntu 创建的软件包。如果要在 Ubuntu 中使用 Cobbler，可以使用“<https://fedorahosted.org/cobbler/wiki/DownloadInstructions>”处提供的指令从源代码进行安装，或者使用第 7 章学过的命令创建一个软件包。

2. 安装软件包

下面开始在 Ubuntu 中设置 PXE 启动服务器并安装部分软件包。需要先安装 TFTP 服务器（用于向目标机中传送文件）以及运行 TFTP 服务器所需的 `inetutils-inetd` 软件包。如果没有安装 DHCP 服务器，则还要安装 `dhcp3-server` 软件包。最后，使用 Apache Web 服务器将发行版本文件发送到要安装的主机上。如果没有安装 Apache 服务器，使用如下命令安装 `apache2`

软件包。

```
$ sudo apt-get install inetutils-inetd tftpd-hpa dhcp3-server apache2
```

■注：在第 9 章中已经讨论过 DHCP 的配置问题。

3. 配置 DHCP 服务器

接下来，需要配置 DHCP 服务器。如果已经有一个配置好了的 DHCP 服务器，建议升级配置文件，为现存的 DHCP 范围指定一个 PXE 启动服务器，否则，需要配置 DHCP 为自动配置主机提供一个适当范围内的地址。例如，可以为自动配置主机创建一个 DHCP 地址范围。

可通过升级配置文件/etc/dhcp3/dhcp.conf 来配置 Ubuntu 的 DHCP 服务器。如列表 19-4 所示，在第 9 章中创建的示例配置文件中添加所需的配置。

列表 19-4 Ubuntu 中示例 DHCP 服务器配置文件

```
allow booting;
allow bootp;
ddns-update-style interim;
ddns-ttl 3600;
default-lease-time 600;
max-lease-time 7200;
log-facility local7;

key dynamic-update-key {
    algorithm hmac-md5;
    secret "3PDRnypPtzJqpbQvbw/B7bhPuHqpUe0Sdi95Z4Ez/IzhS6ldzcK6MJ6CdFHkkegp
TN1kmXOM6GggRNE24aPmOw==";
}

zone 0.168.192.in-addr.arpa. {
    key dynamic-update-key;
    primary 192.168.0.1;
}

zone example.com. {
    key dynamic-update-key;
    primary 192.168.0.1;
}

subnet 192.168.0.0 netmask 255.255.255.0 {
    filename "pxelinux.0";
    next-server 192.168.0.1;
    option routers 192.168.0.254;
    option domain-name "example.com";
    option domain-name-servers 192.168.0.1;
    option broadcast-address 192.168.0.255;
    group "static" {
        use-host-decl-names on;
        host au-mel-rhel-1 {
            hardware ethernet 00:16:3E:15:3C:C2;
            fixed-address au-mel-rhel-1.example.com;
        }
    }
}

pool {
    range 192.168.0.101 192.168.0.150;
```



```

        deny unknown clients;
    }
    pool {
        range 192.168.0.151 192.168.0.200;
        allow unknown clients;
        default-lease-time 7200;
        max-lease-time 21600;
    }
}

```

其中，在文件开始为 `bootp` 和 `booting` 添加了 `allow` 指令，告诉 DHCP 服务器接受网络启动请求。在 `subnet` 指令中，添加了两个选项 `filename` 和 `next-server`。

`filename` 选项指定了 DHCP 服务器给需要网络启动的主机所传递的文件。此文件包含了启动主机的初始化指令，还需要在此处指定 `pxelinux.o` 文件。稍后将安装此文件。

`next-server` 选项告诉网络启动主机从哪个服务器中获取启动文件。在这里要指定 PXE 服务器的 IP 地址。

提示：更改 DHCP 配置文件后，需要重启 DHCP 服务器。

4. 配置 TFTP 服务器

已经安装了包含 TFTP 服务器的 `tftp-hpa` 软件包。现在需要配置此服务器以便于将启动文件传送到目标主机。编辑文件 `/etc/default/tftp-hpa` 来完成配置。

```

#Defaults for tftpd-hpa
RUN_DAEMON="yes"
OPTIONS="-l -s /var/lib/tftpboot"

```

在后面的代码中，通过将其设置为“yes”来启用 `RUN_DAEMON` 选项。注意在 `OPTIONS` 行中指定的 `-l` 和 `-s` 选项。`-l` 选项指定守护进程运行在独立监听模式。`tftpd` 服务通常运行在 `inetd` 下，`inetd` 是一个守护进程管理器，`-l` 选项告诉它将其设置为一个普通守护进程。`-s` 选项告诉守护进程启动文件的位置，即 `/var/lib/tftpboot`，此目录中保存了所有的启动文件。

配置好 `tftpd-hpa` 软件包后，需要重启 `tftpd-hpa` 服务。

```
$ invoke-rc.d tftpd-hpa restart
```

5. 安装启动文件

现在需要安装启动主机所需的文件。这些文件应该安装在 `/var/lib/tftpboot` 目录下。Ubuntu 自带了专为网络启动设计的启动和内核文件集，这些文件都在 Ubuntu 安装媒体的 `install/netboot` 目录下。

```
$ sudo cp -r /media/cdrom/install/netboot/* /var/lib/tftpboot/
```

或者也可以从 Ubuntu 的在线库文件中下载。

```

$ cd /tmp
$ lftp -c "open http://archive.ubuntu.com/ubuntu/dists/hardy/main/installer-i386/
current/images/; mirror netboot/"
$ mv netboot/* /var/lib/tftpboot
$ rm -fr netboot

```

上面使用 `lftp` 命令将 Ubuntu 8.04 或 Hardy 版的网络启动文件从 `netboot` 目录下复制到主

机的/tmp 目录下，然后再将此目录下的内容移动到/var/lib/tftpboot 目录中。

■注：lftp 命令是 FTP 文件传输命令的一个较复杂版本。其工作工程参见命令的 man 页面。

6. 配置 PXE 启动加载器

现在已经有了网络启动文件，接下来可以为主机配置启动加载器了。以下是/var/lib/tftpboot 目录下的内容。

```
boot.img.gz
mini.iso
netboot.tar.gz
pxelinux.0 -> ubuntu-installer/i386/pxelinux.0
pxelinux.cfg -> ubuntu-installer/i386/pxelinux.cfg
ubuntu-installer
```

目录中包含了 5 个文件（其中两个是符号链接）和一个名为 ubuntu-installer 的目录。pxelinux.0（此文件配置到了 DHCP 服务器配置文件的 filename 选项中）、boot.img.gz，和 netboot.tar.gz 文件将完成主机的初始化启动工作。mini.iso 文件是启动文件的 ISO（可刻录到 CD 中）镜像。如果不使用基于网络的自动化配置，可以将此镜像刻录到 CD 中，用它来启动主机并运行一个小型安装环境。pxelinux.cfg 目录中包含了 PXE 用来选择启动镜像的配置文件（稍后将针对主机环境对其加以调整），ubuntu-installer 目录中包含了 PXE 启动所需文件。

查看 pxelinux.cfg 目录，可以找到一个名为 default 的文件，这是主机启动时的默认配置文件，其中内容如下所示。

```
DISPLAY ubuntu-installer/i386/boot-screens/boot.txt

F1 ubuntu-installer/i386/boot-screens/f1.txt
F2 ubuntu-installer/i386/boot-screens/f2.txt
F3 ubuntu-installer/i386/boot-screens/f3.txt
F4 ubuntu-installer/i386/boot-screens/f4.txt
F5 ubuntu-installer/i386/boot-screens/f5.txt
F6 ubuntu-installer/i386/boot-screens/f6.txt
F7 ubuntu-installer/i386/boot-screens/f7.txt
F8 ubuntu-installer/i386/boot-screens/f8.txt
F9 ubuntu-installer/i386/boot-screens/f9.txt
F0 ubuntu-installer/i386/boot-screens/f10.txt

DEFAULT install

LABEL install
    kernel ubuntu-installer/i386/linux
    append ks=http://192.0.2.161/ks.cfg vga=normal initrd=ubuntu-installer/i386/
initrd.gz --
LABEL linux
    kernel ubuntu-installer/i386/linux
    append vga=normal initrd=ubuntu-installer/i386/initrd.gz --
LABEL cli
    kernel ubuntu-installer/i386/linux
    append tasks=standard pkgsel/language-pack-patterns= pkgsel/install-language-
support=false vga=normal initrd=ubuntu-installer/i386/initrd.gz --
```



```

LABEL expert
    kernel ubuntu-installer/i386/linux
    append priority=low vga=normal initrd=ubuntu-installer/i386/initrd.gz --
LABEL cli-expert
    kernel ubuntu-installer/i386/linux
    append tasks=standard pkgsel/language-pack-patterns= pkgsel/install-language-
support=false priority=low vga=normal initrd=ubuntu-installer/i386/initrd.gz --

LABEL rescue
    kernel ubuntu-installer/i386/linux
    append vga=normal initrd=ubuntu-installer/i386/initrd.gz rescue/enable=true --

PROMPT 0
TIMEOUT 0

```

文件规定了启动环境，先是使用 **DISPLAY** 选项指定了主机启动时按下每一个功能键（F1 到 F0 选项）将会显示的文本信息。接下来使用 **LABEL** 选项指定了可以运行的各种启动类型。**DEFAULT** 选项指定了默认的启动标签，在这里默认启动项为 **install**。

每个标签指定了要加载的内核及要传递给内核的选项。

提示：在如下链接处可以找到启动选项相关的文档，见“<https://help.ubuntu.com/community/BootOptions>”。

最后，使用 **PROMPT** 和 **TIMEOUT** 选项控制启动顺序。若将 **PROMPT** 选项设为 1，意味着在启动提示符下等待，直到选定一个启动标签。



图 19-3 Ubuntu 启动提示符

若 **PROMPT** 选项设为 0，启动进程将不等待按键而立即自动选中默认启动选项。**TIMEOUT** 以秒为单位指定了要在启动提示符下等待多长时间。若到时没有选中任何选项，将启动默认标签。若要自动启动并安装主机，则应将 **PROMPT** 的值设为 0。

与其使用默认配置选项，不如在此目录中为要启动和安装特定的主机指定特定的配置文件。可以使用目标主机的 **MAC** 地址为名称创建一个文件。比如，如果主机的 **MAC** 地址是

“00: 0A:E4:2E:A6:42”，可以复制默认文件，编辑后命名为 000AE42EA642。当拥有此 MAC 地址的主机请求网络启动时，将会使用此配置文件启动而不是 default 文件。

7. 为自动化配置 Apache

接下来，将要安装的发行版的内容添加到一个目录中并允许使用 Apache Web 服务器访问。启动进程结束后，将会用到这些文件来安装新的主机。

在目录/var/www 下创建一个名为 Ubuntu 的子目录，然后从 Ubuntu 安装媒体中复制 Ubuntu 的安装文件到其中。

```
$ sudo mkdir /var/www/ubuntu
$ sudo cp -r /media/cdrom/* /var/www/ubuntu/
```

也可以直接挂载 CD 或 DVD。使用 mount 命令挂载媒体，同时使用--bind 选项使媒体文件挂载到目录/var/www/ubuntu 中。这就意味着用户不用将安装文件复制到 PXE 启动主机上。下述命令将挂载到/media/cdrom 下的任何 CD 或 DVD 媒体重新挂载到/var/www/ubuntu 中。

```
$ sudo mount --bind /media/cdrom/ /var/www/ubuntu/
```

然后 Apache 服务器将使用 HTTP 把这些文件发布出去。在 Kickstart 配置文件中，将会告诉主机安装发行版时如何找到这些文件。

■注：第 11 章中讨论过如何配置和运行 Apache 服务器。

8. 防火墙设置

需要配置防火墙以允许网络启动请求访问。为 TFTP 服务器打开 69 端口，同时为 HTTP 传输打开 80 端口。如下是一些适当的 iptables 规则。

```
-A Firewall-eth0-INPUT -s 192.168.0.0/24 -p udp -m state --state NEW --dport 69 ➡
-j ACCEPT
-A Firewall-eth0-INPUT -p tcp -m state --state NEW --dport 80 -j ACCEPT
```

这些规则允许在子网 192.168.0.0/24 中的任何一台主机在适当的端口访问启动服务器。第 6 章有关于防火墙的更多信息。

9. 指定 Kickstart 文件

为完成此配置，需要创建并指定一个 Kickstart 文件来自动运行实际的安装过程。应该将此文件保存在/var/www/ubuntu 目录下并使它对 Apache Web 服务器为可用，就像发行版文件一样。现在先创建一个文件。

```
$ sudo touch /var/www/ubuntu/ks.cfg
```

此命令在/var/www/ubuntu 目录下创建了一个名为 ks.cfg 的文件。

■注：如果挂载了一个 DVD 目录（如前所述），则不能再在/var/www/ubuntu 下保存 ks.cfg 文件，这时应该将此文件置于其他位置（比如，另外创建一个目录来存放此文件）。

下面来编辑 `ks.cfg` 文件并在其中包含一些最基本的配置。

```
install
url --url http://192.168.0.1/ubuntu/
```

示例文件中有两个 Kickstart 选项，即 `install` 和 `url`。`install` 选项告诉 Kickstart 安装一个系统，而不是 `upgrade`，即升级一个现存的系统。`url` 选项告诉 Kickstart 从何处寻找发行版安装文件，这些也就是使用 Apache Web 服务器所发布的文件。

注：当使用此示例文件启动一个主机时，将启动安装进程，但由于没有为安装问题制定任何答案，用户还是需要手动回答每一个问题。下一节将讨论如何为这些问题添加答案。

要使用 `ks.cfg` 配置文件，则需要告诉启动服务器到何处寻找此文件。可以在 PXE 配置文件，如 `/var/lib/tftpboot/pxelinux.cfg/default` 中为一个或多个标签添加一个附加的选项。下面是默认文件中的安装标签内容。

```
LABEL install
    kernel ubuntu-installer/i386/linux
    append ks=http://192.168.0.1/ubuntu/ks.cfg vga=normal
initrd=ubuntu-installer/i386/initrd.gz --
```

可以看到，此处在内核启动选项中添加了一个 `ks` 选项。此 `ks` 选项告诉启动服务器到何处寻找 Kickstart 文件，示例中文件在 `http://192.168.0.1/ubuntu/ks.cfg`。

10. 网络启动一个 Ubuntu 主机

以上已经设置了 Ubuntu 启动服务器并指定了发行版安装文件，现在可以启动并安装主机了。通过进行合适的 BIOS 配置，设置从网络启动主机。

启动的主机将会试着从 DHCP 服务器处获取一个 IP 地址。如果获取成功，主机将请求启动指令。DHCP 服务器将会提供 `pxelinux.0` 文件并将启动主机引导至 PXE 服务器。PXE 服务器将提供合适的启动文件并显示 Ubuntu 的启动屏幕，如图 19-4 所示。



图 19-4 Ubuntu 启动屏幕

主机启动后，将会获取 Kickstart 配置文件并初始化安装进程。下一节将介绍如何自动化此进程。

11. Ubuntu 网络启动排错

网络启动进程排错需要监控主机上的几个文件（包括日志文件），同时使用如 tcpdump 或 tshark 命令的网络监控工具。

下面从查看/var/log/daemon.log 文件以监控 DHCP 进程的输出信息开始。还可以使用一些网络监控工具对主机和启动服务器间的网络通信进行监控。

```
$ sudo tshark port tftp
Capturing on eth0
0.000000 192.168.0.1 -> 192.168.0.161 TFTP Read Request, File: pxelinux.0\000, ➡
Transfer type: octet\000
0.126924 192.168.0.1 -> 192.168.0.161 TFTP Read Request, File: pxelinux.0\000, ➡
Transfer type: octet\000
0.238396 192.168.0.1 -> 192.168.0.161 TFTP Read Request, File: ➡
pxelinux.cfg/564ddfdb-733c-708f-9d49-fa93213b2246\000, Transfer type: octet\000
0.242177 192.168.0.1 -> 192.168.0.161 TFTP Read Request, File: ➡
pxelinux.cfg/01-00-0c-29-3b-22-46\000, Transfer type: octet\000
0.244886 192.168.0.1 -> 192.168.0.161 TFTP Read Request, ➡
File: pxelinux.cfg/C00002A4\000, Transfer type: octet\000
0.246654 192.168.0.1 -> 192.168.0.161 TFTP Read Request, ➡
File: pxelinux.cfg/C00002A\000, Transfer type: octet\000
0.248279 192.168.0.1 -> 192.168.0.161 TFTP Read Request, File: ➡
pxelinux.cfg/C00002\000, Transfer type: octet\000
0.289393 192.168.0.1 -> 192.168.0.161 TFTP Read Request, File: ➡
pxelinux.cfg/C0000\000, Transfer type: octet\000
0.306368 192.168.0.1 -> 192.168.0.161 TFTP Read Request, File: ➡
pxelinux.cfg/C000\000, Transfer type: octet\000
0.311124 192.168.0.1 -> 192.168.0.161 TFTP Read Request, File: ➡
pxelinux.cfg/C00\000, Transfer type: octet\000
0.315184 192.168.0.1 -> 192.168.0.161 TFTP Read Request, File: ➡
pxelinux.cfg/C0\000, Transfer type: octet\000
0.318966 192.168.0.1 -> 192.168.0.161 TFTP Read Request, File: ➡
pxelinux.cfg/C\000, Transfer type: octet\000
0.333135 192.168.0.1 -> 192.168.0.161 TFTP Read Request, File: ➡
pxelinux.cfg/default\000, Transfer type: octet\000
0.355684 192.168.0.1 -> 192.168.0.161 TFTP Read Request, File: ➡
ubuntu-installer/i386/boot-screens/boot.txt\000, Transfer type: octet\000
0.361184 192.168.0.1 -> 192.168.0.161 TFTP Read Request, File: ➡
ubuntu-installer/i386/boot-screens/splash.rle\000, Transfer type: octet\000
```

这里使用 tshark 命令（使用 tshark 软件包安装）来监控在 69 端口（TFTP 端口）上的所有网络流。可以看到对文件 pxelinux.0 以及配置文件 pxelinux.cfg/default 的请求，最后显示了 Ubuntu 启动屏幕。

■注：关于网络启动的更多信息，参见 “<https://help.ubuntu.com/8.04/installation-guide/i386/install-methods.html>”。

19.1.3 Kickstart 和 Preseed

在 Red Hat 上，用来自动安装主机的语言称为 Kickstart。在 Ubuntu 中，称为 Preseed。然而，Ubuntu 也支持 Kickstart，只不过变成了 Kickseed。目前的 Kickseed 支持不能完全配置 Ubuntu，不过可以使用精选的 Preseed 指令来解决中间的差距。为了简单起见，也因为这是最简单的语言，还是使用 Kickstart 来进行 Red Hat 和 Ubuntu 的自动化安装配置。若其中出现不支持 Ubuntu 之处，也可以使用 Preseed 来进行配置。

■注：Kickstart 对 Ubuntu 的支持正在不断加强。8.10 版和 Jaunty 版（比如，最新的 9.04 版）将更加增强其支持性。

Kickstart 配置文件中包含了自动化安装进程所需的指令。它是一个最简单的安装选项脚本过程，但也可以将其扩展为一个复杂的配置。Kickstart 在 Red Hat 中使用较多，最近也开始在 Ubuntu 中使用，因此有详实的文档。可以在“https://www.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5/html/Installation_Guide/ch-kickstart2.html”处的 RHEL 5 中找到 Kickstart 的详尽文档。此手册对于 Ubuntu 主机的自动化配置也是很有用的，因为在其中用到的指令很多都是一样的。同样，如果特定指令不支持 Ubuntu 时，将提供 Preseed 中的类似功能。

可以在“<https://help.ubuntu.com/8.04/installation-guide/i386/appendix-preseed.html>”处找到 Preseed 及其指令的文档。本节稍后将介绍其中的部分指令。

上面介绍了如何在自动化安装环境中指定 Kickstart 文件，在 Red Hat 中使用 Cobbler，而在 Ubuntu 中使用 PXE 启动配置。列表 19-5 中列出了示例 Kickstart 文件中的部分内容。

列表 19-5 一个 Kickstart 文件

```
install
# System authorization information
auth --useshadow --enablemd5
# System bootloader configuration
bootloader --location=mbr
# Partition clearing information
clearpart --all --initlabel
# Use text mode install
text
```

列表 19-5 显示了一个以 install 选项开始的配置指令列表，将此安装的行为定义为一个安装过程。另一个选项是 upgrade，此选项将为主机启动一个自动升级过程。

接下来是带有选项的指令，例如，“auth --useshadow --enablemd5”告诉 Kickstart 如何回答特定的安装问题。auth 声明项带有两个值--useshadow 和--enablemd5，分别启用了影子密码和 MD5 密码验证。

接下来的选项，bootloader 带有值“--location=mbr”，这告诉 Kickstart 将启动加载器安装到 MBR。下面是指令 clearpart，此项清除了主机中的所有部分并为其创建一个默认的标签。最后一个选项 text 指定了需要使用基于文本的安装过程而不是 GUI。

■提示：可以使用 Kickstart 来升级或安装一个主机。如果有一个现存的主机，可以从操作系统的新版本来进行网络启动，并使用 Kickstart 文件编辑脚本升级系统。

有太多指令，因此不便于一一讨论。在表 19-1 中列出了必须指定的指令和其他后面可能有用的主要指令。

表 19-1 所需的 Kickstart 指令	
指令	描述
auth	配置认证方式
bootloader	配置启动加载器
keyboard	配置键盘类型
lang	配置主机语言
part	配置分区。安装时需要，升级时不需要
rootpw	指定 root 用户的密码
timezone	指定主机所在的时区

还可以在如下链接处找到有用的指令列表及其解释，见 http://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/Installation_Guide-en-US/s1-kickstart2-options.html。

■提示：如果使用的是 Red Hat 平台，在/root/anaconda-ks.cfg 文件中安装主机后，将会创建一个示例 Kickstart 文件。此文件将会展示目前的主机是如何创建的，并可以用来作为创建类似主机的示例。

1. 安装源文件

读者已经见到了 install 和 upgrade 指令是如何指定安装类型的，还可以指定安装的源文件。在 Ubuntu 自动化安装部分，有 url 指令，可以告诉 Kickstart 从一个 HTTP URL（比如，<http://192.168.0.1/ubuntu>）处获取安装文件。

```
url --url http://192.168.0.1/ubuntu/
    对于 Cobbler，可以定义一个变量类指定安装源文件的位置。
```

```
url --url=$tree
    url 指令也可以用来指定一个 FTP 服务器。
url --url ftp://jsmith:passsword@192.168.0.1/ubuntu
```

当从本地挂载的 CD 或 DVD 以及本地分区的硬盘驱动器安装时，可以使用包括 cdrom 在内其他指令指定一个替换源文件。

```
harddrive --dir=/ubuntu --partition=/installsource
```

2. 键盘、语言和时区

这个小节简单讨论键盘、语言和时区的配置。

```
# System keyboard
keyboard us
```



```
# System language
lang en_AU
# System timezone
timezone Australia/Melbourne
```

这里指定 `us` 作为 `keyboard` 指令的值以指定使用 US 键盘。同时指定语言为 `en_AU`（澳大利亚英语），时区为 `Australia/Melbourne`。

■提示：在 Red Hat 和 Ubuntu 中的键盘、语言和时区是一样的。

3. 管理用户

还可以使用 Kickstart 的 `rootpw` 指令设置 root 用户的密码。

```
rootpw --iscrypted $1$V.rhw$VUj.euMxoV9WkcQSanpGi0
```

`rootpw` 指令是所有 Kickstart 文件中都必需的一个 Kickstart 选项。可以为其指定一个明文文本值，当指定 “`--iscrypted`” 选项时还可以为 root 用户指定一个加密的密码值。可以使用如下 `grub-md5-crypt` 命令来创建一个加密的密码。

```
$ grub-md5-crypt
Password:
Retype password:
$1$V.rhw$VUj.euMxoV9WkcQSanpGi0
```

指定一个想要加密的密码然后按照提示重新输入一次密码，然后剪切生成的值并将加密过的密码粘贴到 Kickstart 文件中。

在 Ubuntu 中，`rootpw` 指令默认使用 “`--disabled`” 选项，保持 Ubuntu 禁止使用 root 用户。

```
rootpw --disabled
```

■注：在 Red Hat 中不存在 “`--disabled`” 选项。

在 Ubuntu 中，Kickstart 还可以使用 `user` 指令创建一个新用户。

```
user jsmith --fullname "John Smith" --password password
```

上面的代码创建了一个名为 `jsmith` 的新用户，其全名为 `John Smith`，密码是 `password`。添加了 “`--iscrypted`” 选项后，可以为用户添加一个加密的密码。可以使用 `rootpw` 指令为用户创建一个加密密码。

4. 防火墙和网络

在 Red Hat 中，可以如下设置初始化防火墙和网络配置。

```
# Firewall configuration
firewall --enabled --http --ssh --smtp
# SELinux configuration
selinux --disabled
```

上面使用 `firewall` 选项启用了防火墙并允许通过 HTTP、SSH 和 SMTP 进行访问（可以使用 `disabled` 选项停用防火墙）。同时，使用 “`selinux--disabled`” 选项停用了 SELinux。

Ubuntu 不能通过 Kickstart 或 Preseed 来修改防火墙配置，因此只能手动进行如下设置。

```
firewall --disabled
```

在 Red Hat 和 Ubuntu 中，都可以使用 Kickstart 配置网络连接，如下所示。

```
# Network information
network --bootproto=static --device=eth0 --gateway=192.168.0.254 ➡
--ip=192.168.0.1 --nameserver=192.168.0.1 --netmask=255.255.255.0 --onboot=on
```

还可以使用 `network` 选项为一个或多个网络接口指定网络配置。上面已经设置了多个选项来配置 `eth0` 接口了。还可以指定 DHCP 配置，如下所示。

```
network --bootproto=dhcp --device=eth0 --onboot=on
```

在 Red Hat 上的 Cobbler 中，如果使用一个特定的主机（使用“`cobbler system`”命令创建的主机），可以将此主机的特定网络配置值传递到 Cobbler 系统配置中去。

```
$ sudo cobbler system edit --name=gateway.example.com --mac=00:0C:29:3B:22:46 ➡
--profile=RHEL5 --interface=0 --ip=192.168.0.1 --subnet=255.255.255.0 -- ➡
gateway=192.168.0.254 --hostname=gateway --static=1
```

上面指定了 `edit` 命令对一个现存的已经配置好了的 Cobbler 系统进行修改并将网络配置值传递到系统中，这将给 `eth0` 接口定义一个静态的网络配置。其中使用“`--static=1`”选项来说明这是一个静态配置，若使用 DHCP 配置则应该使“`--static=0`”。使用“`--interface=0`”来指定要配置的端口号。

然后，没有使用 `network` 行，而是在 Kickstart 文件中使用了 Cobbler 称之为 `snippet` 的行。

```
$SNIPPET('network_config')
```

创建主机时，Cobbler 把定义好的网络配置传递给这个 `snippet` 及其模板，然后转换为适当的 `network` 行，这样主机就配置好了。

■提示：此 `snippet` 是 Cobbler 的 `snippet` 系统的一个简单应用，还可以使用它定义各种各样的动作，可以在 `/var/lib/cobbler/snippets` 目录下找到各种可用选择，其中也包括上面用过的 `network_config` `snippet`。在 `sample.ks` 文件中可以看到 `snippets` 的使用方法，还可以在“<https://fedorahosted.org/cobbler/wiki/KickstartTemplating>”和“<https://fedorahosted.org/cobbler/wiki/KickstartSnippets>”处找到关于如何使用模板和 `snippets` 的命令。

5. 磁盘和分区

前面已经见过一个 Kickstart 用来配置磁盘和分区的选项了，就是 `clearpart`，此选项将清除主机上的分区信息。接下来，可以使用 `part` 选项来配置主机上的分区，如下所示。

```
# Partition clearing information
clearpart --all --initlabel
part /boot --asprimary --bytes-per-inode=4096 --fstype="ext3" --size=150
part / --asprimary --bytes-per-inode=4096 --fstype="ext3" --size=4000
part swap --bytes-per-inode=4096 --fstype="swap" --size=512
```

■注：在 Red Hat 中，可以简单地指定 `autopart` 选项来创建一个类似的配置。`autopart` 选项将自动创建 3 个分区。第一个分区是 1GB 或更大的根（`/`）分区，第二个是交换分区，第三个是基于主机架构的一个适当的启动分区。默认分区的大小都可以使用 `part` 指令进行重新设置。

可以使用 `part` 选项来创建一个特定分区。前面的代码中，首先创建了两个分区 `/boot` 和 `/`，两者都是 `ext3` 格式的。其中为 `/boot` 分配了 150MB 空间，为 `/` 或根分区分配了 4000MB（或 4GB）的空间。同时还创建了一个大小为 512MB 的交换分区。

在 Red Hat 中使用 Kickstart 可以创建软件 RAID 配置，如下所示。

```
part raid.01 --asprimary --bytes-per-inode=4096 --fstype="raid" --grow --ondisk=sda --size=1
part raid.02 --asprimary --bytes-per-inode=4096 --fstype="raid" --grow --ondisk=sdb --size=1
part raid.03 --asprimary --bytes-per-inode=4096 --fstype="raid" --grow --ondisk=sdg --size=1
part raid.04 --asprimary --bytes-per-inode=4096 --fstype="raid" --grow --ondisk=sdd --size=1
part raid.05 --asprimary --bytes-per-inode=4096 --fstype="raid" --grow --ondisk=sde --size=1
raid / --bytes-per-inode=4096 --device=md0 --fstype="ext3" --level=5 raid.01 raid.02 raid.03
raid.04 raid.05
```

上面定义了 5 个 RAID 磁盘，同时使用 “`--grow`” 选项指定每一个磁盘都使用其全部空间创建。

各自使用的磁盘设备分别用 “`--ondisk`” 选项加以指定，这里是从 `sda` 到 `sde`。最后，使用 `raid` 选项指定设备号为 `md0` 的 RAID 磁盘作为 `/` 或根分区。

警告：在 Ubuntu 中，Kickstart 不支持 RAID，Preseed 也不例外。Ubuntu 中 8.04 版的 Preseed 对 RAID 的支持大大增强，但是并不建议读者使用它来配置和自动安装 RAID 主机。

还可以在自动化安装过程中使用 LVM 创建分区。比如，在 Red Hat 中，可以按如下格式进行创建。

```
part /boot --fstype ext3 --size=150
part swap --size=1024
part pv1 --size=1 --grow
volgroup vg_root pv1
logvol / --vgname=vg_root --size=81920 --name=lv_root
```

上述示例中，创建了一个 150MB 的启动分区、一个 1GB 的交换分区和一个使用所有剩余空间的名为 `pv1` 的物理卷，使用 “`--grow`” 选项来填充剩余的磁盘空间。然后创建了一个 80GB 的名为 `vg_root` 的 LVM 逻辑卷。

在 Ubuntu 中，可以使用 `preseed` 指令通过 Preseed 来配置 Kickstart 文件中的 LVM。

```
preseed --owner d-i partman-auto/method string lvm
preseed --owner d-i partman-lvm/device_remove_lvm boolean true
preseed --owner d-i partman-lvm/confirm boolean true
preseed --owner d-i partman-auto/choose_recipe select atomic
preseed --owner d-i partman/confirm_write_new_label boolean true
preseed --owner d-i partman/choose_partition select finish
preseed --owner d-i partman/confirm boolean true
```

`preseed` 指令非常简便，同时也是在 Kickstart 配置文件中包含 Preseed 指令的一种方法。指令格式如下所示。

```
preseed --owner owner key/sub-key value
```

第一部分 `preseed` 告诉 Kickstart 现在使用的是 Preseed 指令。“`--owner`” 选项告诉 Preseed 此指令属于哪一个应用程序或功能模块，比如，“`d-i`” 表示一个 Preseed 指令属于 Debian 安

装器。

■注：若没有指定一个 “--owner” 选项，则 Preseed 默认赋值为 “d-i”。

键/子键的值定义了要设置的 Preseed 选项，这些值的结构就像一个键值集合，每一个键值下面都带有一系列的子键值。例如，partman 键包含了所有的 Preseed 分区配置选项。这里，在此键下，可以看到 confirm 和 confirm_write_label 子键。

```
partman/confirm
partman/confirm_write_new_label
```

type 字段包含了值的类型，比如一个字符串或一个布尔类型。value 字段包含了实际的设置值。

```
preseed --owner d-i partman-auto/method string lvm
```

下面使用 device_remove_lvm 命令来移除所有现存的 LVM 设备，同时使用 confirm 指令忽略此过程中弹出的确认信息。

```
preseed --owner d-i partman-lvm/device_remove_lvm boolean true
preseed --owner d-i partman-lvm/confirm boolean true
```

接下来，使用 partman/choose_recipe 键选择分区的方式，在这里使用 atomic。

```
preseed --owner d-i partman-auto/choose_recipe select atomil
```

atomic 方式将所有的文件创建到一个分区中，这是 Ubuntu 安装中所建议的默认选择。除了 atomic，也可以使用 home 方式指定一个单独的/home 分区，使用 multi 方式指定独立的/home、/usr、/var 和/tmp 分区。

接下来，告诉 Kickstart 自动完成并写入 LVM 配置信息。

```
preseed --owner d-i partman/choose_partition select finish
preseed --owner d-i partman/confirm boolean true
```

■提示：在 Ubuntu 的 9.04 发行版中，将支持本地化 LVM 配置 Kickstart 文件，而不再使用 preseed 指令来替代。

还可以使用在链接 “<https://help.ubuntu.com/8.04/installation-guide/i386/preseed-contents.html>” 处的帮助信息对磁盘和分区进行精确的个性化配置。

■注：在第 8 章中讨论了分区、软件 RAID 以及 LVM 的相关内容。

6. 软件包管理

可以使用 Kickstart 指定要安装的软件包。在 Red Hat 中，可以使用%packages 来开始定义一段，其中包含着要安装的软件包或软件包组列表。

```
%packages
@ Administration Tools
@ Server Configuration Tools
@ System Tools
@ Text-based Internet
dhcp
```


其中每一行都有一个@符号和一个空格，然后就是要安装的软件包组名称，比如，管理工具就是 Administration Tools。也可以只列出名称而没有@符号和空格来指定一个单独的软件包，就像上面的 dhcp 软件包。

Ubuntu 中使用类似的设置。

```
%packages
@ kubuntu-desktop
dhcp-client
```

上面的代码安装了 Kubuntu 桌面软件包和 dhcp-client 软件包。

■注：第 7 章中讨论了软件包组的相关内容。

7. 安装动作

可以对安装过程中的动作进行配置，如下所示。

```
# Run the Setup Agent on first boot
firstboot -disable
# Skip installation key
key --skip
# Reboot after installation
reboot
```

firstboot 和 key 指令都是 Red Hat 专用的，与 Ubuntu 无关。firstboot 指令是否启用通常在首次启动时显示的安装后菜单。key 指令强制输入一个 Red Hat 安装码，可以使用“--skip”选项跳过输入屏幕。

Red Hat 安装码

Kickstart 还支持在系统安装时指定一个 Red Hat 安装码（第 2 章中作过讨论）。可以在 Kickstart 文件中加入如下代码。

```
# RHEL Install Key
key $getVar('rhel_key', '--skip')
```

然后就可以使用 Cobbler 系统命令为每个主机添加 Kickstart 元数据，如下所示。

```
$ sudo cobbler system edit --name=00:0C:29:3B:22:46 ➡
--ksmeta="rhel_key=4ea373203ae2bd77"
```

使用用户自己的主机安装码替换以上的 4ea373203ae2bd77 就可以了。若不想指定安装码，Kickstart 可以使用“--skip”选项跳过输入安装码的应用程序。

Reboot 指令告诉 Kickstart 安装完成后重启主机，也可以使用 shutdown 指令告诉主机安装完成后关机。

8. 安装前与安装后

在 Kickstart 安装主机前后都可以运行一些脚本。安装前运行的脚本都在 Kickstart 配置文件传入之后、完成主机配置之前运行。所有的安装前运行脚本都放在 Kickstart 文件尾部并以 %pre 为前缀。

安装后运行脚本在配置完成和主机安装完后被触发。这些脚本也都放在 Kickstart 文件的尾部并以 %post 行为前缀。


```
%post
$SNIPPET('post_install_kernel_options')
$SNIPPET('post_install_network_config')
$SNIPPET('redhat_register')
```

以上代码指定了 3 个安装后运行的 Cobbler snippet, 用来配置内核和网络选项并使用 Red Hat 注册主机（就像前面使用的 RHN 一样）。

这个安装后运行的脚本空间对与运行任何必须的安装应用程序或脚本程序都是非常有用的。

9. Kickstart 配置

另外一种创建 Kickstart 文件的方式是使用基于 GUI 的 Kickstart 配置器工具（参见 http://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/Installation_Guide-en-US/ch-redhatconfig-kickstart.html）。在 Red Hat 中可以使用 system-config-kickstart 命令来安装此应用程序软件包。

```
$ sudo yum install system-config-kickstart
```

在 Ubuntu 中，可以如下进行安装。

```
$ sudo apt-get install system-config-kickstart
```

然后可以从“应用程序”>“系统工具”>“Kickstart”菜单或使用如下命令来启动 Kickstart 配置器。

```
$ system-config-kickstart
```

图 19-5 显示了应用程序的界面。



图 19-5 Kickstart 配置器界面

如果不想在命令行中进行编辑，使用 Kickstart 配置器通常是快速创建一个简单的 Kickstart 配置文件的简便方法。

10. 完全 Kickstart 配置

到现在为止，讨论了很多 Kickstart（还有 Preseed）配置的片段了。下面来看一个在 Red Hat 和 Ubuntu 中的完整配置示例。

先看 Red Hat 中的一个完整示例。

```
install
reboot
url --url=$tree
key --skip
firstboot --disable
auth --enablemd5 --useshadow
bootloader --loader=mbr
keyboard us
lang en_AU
timezone Australia/Melbourne
rootpw --iscrypted $1$V.rhw$VUj.euMxoV9WkcQSanpGi0
firewall --enabled --http --ssh --smtp
selinux --disabled
network --bootproto=dhcp --device=eth0 --onboot=on
clearpart --all --initlabel
autopart
%packages
@ Administration Tools
@ Server Configuration Tools
@ System Tools
@ Text-based Internet
dhcp
```

这里创建了一个非常简单的安装脚本。运行一次安装并在安装后重启主机。其中配置了一些必须的值，如键盘、语言、时区等，而且为 root 用户创建了一个密码。启用了防火墙并允许对 HTTP、SSH 和 SMTP 端口的访问，同时规定从 eth0 端口上请求一个 DHCP 地址。清除了所有已有的分区，然后使用 autopart 指令将主机上的第一块硬盘自动分区。最后安装了一系列软件包组和 dhcp 软件包。

下面是 Ubuntu 中的一个完整示例。

```
install
reboot
url --url http://192.0.2.161/ubuntu/
auth --useshadow --enablemd5
bootloader --location=mbr
keyboard us
lang en_AU
timezone Australia/Melbourne
rootpw --disabled
user jsmith --fullname "John Smith" --password password
selinux --disabled
firewall --disabled
network --bootproto=dhcp --device=eth0 --onboot=on
```



```
preseed --owner d-i partman-auto/method string lvm
preseed --owner d-i partman-lvm/device_remove_lvm boolean true
preseed --owner d-i partman-lvm/confirm boolean true
preseed --owner d-i partman-auto/choose_recipe select atomic
preseed --owner d-i partman/confirm_write_new_label boolean true
preseed --owner d-i partman/choose_partition select finish
preseed --owner d-i partman/confirm boolean true
%packages
@ kubuntu-desktop
dhcp-client
```

Ubuntu 中的配置和 Red Hat 中的很相似，但也针对此发行版进行了适应性个性化配置，比如，使用 `rootpw` 指令禁用了 `root` 用户。还可以看到使用 `preseed` 指令直接指定了一些 Preseed 选项来自动化配置主机中的 LVM。

■注：本节中讨论的两个 Kickstart 完整示例文件在本书的源代码中都有，可以在 Apress 网站 (<http://www.apress.com>) 的源代码 (Source Code) 部分找到。

19.2 配置管理

目前，本书已经向读者展示了配置 Linux 服务器包括相当多的任务，比如，配置主机，创建用户以及管理应用程序、守护进程和服务。为添加新的配置或修复一个由于错误、崩溃或开发的原因而发生改变的配置，这些任务在主机的生命周期中可能要重复多次。这将会是非常费时的，而且通常会造成时间和精力上的极大浪费。

对这个问题的第一反应往往是把这些任务自动化，因而衍生出了个性化配置脚本和应用程序的开发。但是针对此专案开发的脚本却鲜有发布、记录或重用的，因此同样的工具就被一次又一次地重复开发着。同时这些脚本的扩展性极差，而且通常需要不断地进行维护。

配置管理工具可以高效地自动完成这些任务，同时给予主机一个稳定的、可重复的生命周期。下面将讨论如何使用其中的一个工具 Puppet 来完成主机的自动化配置。

19.2.1 Puppet 简介

Puppet (<http://reductivelabs.com/>) 是一个开源的配置管理工具，它基于客户端/服务器端部署模型。它使用 GPLv2 许可证授权。下面将简单介绍 Puppet 的概况以及如何用它配置应用环境和主机。

使用 Puppet 时，先安装和配置中央服务器，也叫 Puppet master。然后在要管理的目标主机上安装客户端软件，也称为 puppet 或 node。配置文件将会在 Puppet 主机上定义及编译，并在 Puppet 客户端连接时将这些配置应用到客户端中。

Puppet 使用运行在 TCP 端口 8140 的 HTTPS 上的 XML-RPC 网络服务提供客户端/服务器连接。使用内部生成的自签名证书和加密的会话来提供安全性。每一个 Puppet 客户端都会产生一个自签名的证书以便于在 Puppet 主机上进行验证和认证。

此后，每个客户端默认每隔 30 分钟连接一次服务器以确定其配置是最新的，此时间间隔可以进行修改。若有一个新的配置文件或原来的配置文件已经修改，则会立即编译并应用到客户机中。如果有必要，则可以在服务器上触发一个配置更新进程，强制将配置文件应用到客户端。客户端中的任何配置发生了变化都会使用从服务器上来的原始配置文件对其进行纠正。任何活动的结果都会被记录并传送到服务器中。

Puppet 工作的中心思想就是一种允许用户阐述和表达配置内容的语言。各个配置组件组织到称为 `resources` 的条目中，然后再组织成 `collections`。其中资源包括如下内容。

- 类型
- 标题
- 属性

列表 19-6 显示了一个示例资源文件。

列表 19-6 一个 Puppet 资源文件

```
file { "/etc/passwd":  
    owner => "root",  
    group => "root",  
    mode => 0644,  
}
```

列表 19-6 中的所示的资源是一个 `file` 类型的资源。`file` 资源配置了所管理的文件的属性。此例中，资源配置了 `/etc/passwd` 文件，将其所有者和用户组都设置为 `root` 用户，并将访问权限设置为 `0644`。

资源的类型告诉 Puppet 正在配置的是何种类型的资源，例如，`user` 和 `file` 类型分别用来管理节点上的用户和文件操作。Puppet 自带了很多默认的资源类型，包括用来管理文件、服务、软件包、`cron` 作业以及文件系统的类型和其他类型。

■提示：可以在“<http://reductivelabs.com/trac/puppet/wiki/TypeReference>”找到一个内置资源类型的完整列表。用户也可以使用 Ruby 编程语言开发自己的类型。

资源的标题作为其在 Puppet 中的标识。每一个标题以资源类型（比如 `file`）和资源的名称（比如 `/etc/passwd`）构成，两个值结合组成资源的标题（比如，`File["/etc/passwd"]`）。

■注：在资源的标题中，资源的类型首字母大写（`File`），资源的名称包含在中括号和双引号中。

资源的名称 `/etc/passwd` 还告诉 Puppet 要管理的文件路径。由 Puppet 管理的每一个文件必须唯一，比如，只能有一个名为“`File["/etc/passwd"]`”的资源。

资源的属性描述了要管理的配置详情，比如定义一个特定的用户及用户的属性（例如，用户所属的组或用户主目录的位置）。列表 19-6 中管理了文件的所有者、组和访问模式（或权限）属性。每一个属性都用 `=>` 符号与其值隔开并以逗号结束。

Puppet 还使用了集合的概念，可以使用集合将很多资源组合到一起。例如，类似于 Apache 的应用程序由软件包、服务和很多配置文件组成。在 Puppet 中，每一个组成部分都表示一种资源（或一个资源组），随后可将其组合在一起并应用到一个节点。本章稍后会简单介绍一下

其中的一些类型。

19.2.2 安装 Puppet

下面开始安装 Puppet。Puppet 的服务器安装和客户端安装只有些微的不同，稍后会分别加以介绍。

1. Red Hat 中的安装

在 Red Hat 中，无论是服务器端还是客户端，都要先安装一些先决条件，其中就包括 Ruby 编程语言。

```
$ sudo yum install ruby ruby-shadow
```

接下来，在主机中添加 EPEL 库文件并从中安装几个软件包。如果此库文件没有添加到 Yum 配置中，可以通过添加 epel-release RPM 包来添加 EPEL 库文件。

```
$ sudo rpm -Uvh http://download.fedora.redhat.com/pub/epel/5/i386/epel-release-5-3.noarch.rpm
```

在服务器或主机上，通过 EPEL 库文件安装 puppet、puppet-master 和 facter 软件包。

```
$ sudo yum install puppet puppet-server facter
```

puppet 软件包中是客户端软件，puppet-master 软件包中是服务器软件，facter 软件包中包含了一个名为 Facter 的系统清单工具。Facter 收集主机的信息或事实，用来定制 Puppet 配置。

在客户端，只需安装 puppet 和 facter 软件包。

```
$ sudo yum install puppet facter
```

2. Ubuntu 中的安装

Ubuntu 中，必须安装的软件包是 puppet、puppetmaster 和 facter。puppet 软件包中是 Puppet 客户端软件，puppetmaster 软件包中是主机端软件，facter 软件包中包含了系统清单工具 Facter。

在服务器或主机端，需要安装：

```
$ sudo apt-get install puppet puppetmaster facter
```

在客户端，需要安装：

```
$ sudo apt-get install puppet facter
```

■注：安装 puppet、puppetmaster 和 facter 软件包后也将同时安装一些必须的软件包。

19.2.3 配置 Puppet

下面从设置 Puppet 主机开始配置 Puppet。配置文件，包括 manifests（包含本机配置的文件），都保存在/etc/puppet 目录下。Puppet 的基本配置文件位于/etc/puppet/puppet.conf。

实际的配置文件将保存在/etc/puppet 目录下的名为 manifests 的目录中。此路径为 puppet

软件包安装时所创建。`manifests` 目录需要包含一个名为 `site.pp` 的文件，此文件是配置的根文件。现在就来创建它。

```
$ sudo touch /etc/puppet/manifests/site.pp
```

■注：包含配置信息的 `manifest` 文件都有一个 `.pp` 后缀。

还要创建 3 个目录，即 `classes`、`nodes` 和 `files`，这 3 个文件中将包含附加的配置信息。

```
$ sudo mkdir /etc/puppet/manifests/{classes,files,nodes}
```

`files` 目录包含了所有要发送到需配置的主机的文件。`nodes` 目录包含了客户机或节点的定义信息。`classes` 目录中包含了各种类。`classes` 是资源的集合，例如，一个 `Apache` 类包含了所有配置 `Apache` 所需的资源。

下面通过在 `site.pp` 文件中定义这些新的目录继续配置 `Puppet`，如列表 19-7 所示。

列表 19-7 `site.pp` 文件

```
import "nodes/*.pp"
import "classes/*.pp"
```

```
$puppetserver = "puppet.example.com"
```

`import` 声明告诉 `Puppet` 将 `nodes` 目录和 `classes` 目录中所有带有 `.pp` 后缀的文件都加载到 `Puppet` 中。`$puppetserver` 声明设置了一个变量。在 `Puppet` 中，由美元符号（\$）开始的配置声明都是变量，可以用来在 `Puppet` 配置中赋值。

在列表 19-7 中，创建了一个包含 `Puppet` 服务器标准域名的变量，并使用双引号将其包含起来。

■注：`Puppet` 中的引用规则参见 <http://reductivelabs.com/trac/puppet/wiki/LanguageTutorial#quoting>。

建议读者为 `Puppet` 主机创建一个 DNS CNAME（比如，`puppet.example.com`）或将其添加到 `/etc/hosts` 文件中。

```
# /etc/hosts
127.0.0.1 localhost
192.168.0.1 au-mel-ubuntu-1 au-mel-ubuntu-1.example.com puppet puppet.example.com
```

■注：在第 9 章中介绍了如何创建 CNAME。

还要在 `/etc/puppet.conf` 配置文件中指定标准域名。配置文件被分成了几个部分，每个部分配置了 `Puppet` 中的一个特定元素。例如，`[puppetd]` 部分配置了 `Puppet` 的客户端，`[puppetmasterd]` 部分则配置了 `Puppet` 的主机或称为服务器。现在只需在其中加入一个条目 `certname` 作为此文件的开始。将此条目的值添加到 `[puppetmasterd]` 部分中（如果在文件中还没有此部分，可以手动创建一个）。

```
[puppetmasterd]
certname=puppet.example.com
```

■注：使用用户主机的标准域名代替 “`puppet.example.com`”。

在很多 Ubuntu 主机和 Red Hat 主机中，添加 `certname` 选项会解决一个由 Ruby SSL 代码带来的错误。关于此错误的更多详细信息，请参见“<http://reductivelabs.com/trac/puppet/wiki/RubySSL-2007-006>”。

1. 设置 Puppet 文件服务

除了要配置各种各样的资源外，Puppet 还可以提供文件服务，比如，它可以把配置文件传送到一个节点中。文件服务器使用 `/etc/puppet/fileserver.conf` 配置文件进行配置。列表 19-8 中是本文件的一个示例。

列表 19-8 `fileserver.conf` 配置文件

```
[files]
path /etc/puppet/manifests/files
allow 192.168.0.0/24
allow 127.0.0.1
```

文件服务器的配置很简单。指定一个名为 `files` 的文件共享并将其包含在中括号[]中。然后，指定文件共享的 `path`，此处是前面已经创建的一个目录 `/etc/puppet/manifests/files`。最后指定 `allow` 和/或 `deny` 声明来控制对文件共享的访问。以上配置中允许了子网 `192.168.0.0/24` 中的任何主机以及来自本机 `127.0.0.1` 的共享文件访问。

■提示：更多文件服务的内容，参见“<http://reductivelabs.com/trac/puppet/wiki/FileServingConfiguration>”。

2. Puppet 防火墙配置

Puppet 主机运行在 TCP 的 8140 端口上。需要在主机的防火墙中开启此端口，客户端也必须能够路由和连接到此主机。要实现这一点，需要在主机中应用适当的防火墙规则，如下所示。

```
-A Firewall-eth0-INPUT -p tcp -m state --state NEW --dport 8140 -j ACCEPT
```

上面的代码允许了任何客户端到 TCP8140 端口的访问。

3. 启动 Puppet 服务器

可以使用 `init` 脚本启动 Puppet 主机。在 Red Hat 中，使用 `service` 命令来运行 `init` 脚本，如下所示。

```
$ sudo service puppetmaster start
```

在 Ubuntu 中，使用 `invoke-rc.d` 命令运行。

```
$ sudo invoke-rc.d puppetmaster start
```

Red Hat 主机中守护进程的输出信息参见 `/var/log/messages`，Ubuntu 主机中的参见 `/var/log/daemon.log`。

19.2.4 连接第一个客户端

一旦配置好 Puppet 主机并启动后，可以配置并初始化第一个客户端。前面提到过，在客

户端中需要使用发行版软件包管理系统安装 puppet 和 facter 软件包。接下来可以在 gateway.example.com 主机上安装一个客户端并连接到 puppet.example.com 主机上。此安装将创建一个带有 puppet.conf 配置文件的/etc/puppet 目录。

连接到客户端后，首先应该从命令行运行 Puppet 客户端，而不是作为一个服务来运行，这样可以看到连接时所发生的一切。Puppet 客户端二进制文件称为 puppetd，在列表 19-9 中可以看到已经初始化了一个到主机的连接。

列表 19-9 Puppet 客户端链接到 Puppet 主机

```
gateway$ puppetd --server=puppet.example.com --no-daemonize --verbose
info: Creating a new certificate request for gateway.example.com
info: Creating a new SSL key at /var/lib/puppet/ssl/private_keys/gateway.example.com
.pem
warning: peer certificate won't be verified in this SSL session
notice: Did not receive certificate
```

在列表 19-9 中，运行了带有多个选项的 puppetd 二进制文件。第一个选项 “--server” 指定了要连接到的 Puppet 主机名称或地址，也可以在客户端的/etc/puppet/puppet.conf 配置文件中在 main 部分指定。

```
[main]
server=puppet.example.com
```

“--no-daemonize” 选项使 Puppet 客户端在前台运行并阻止其以默认配置作为守护进程运行。“--verbose” 选项启用客户端详细输出。

■提示：“--debug” 选项提供了可用于排错的更多信息。

在列表 19-9 中，可以看到连接输出的信息。客户端创建了一个证书签名请求和一个私有密钥以确保链接的安全性。Puppet 使用 SSL 证书对主机和客户端间的连接进行验证。客户端等待主机签署证书允许连接。此时，客户端仍在运行并等待证书签名。客户端将会每隔 2 分钟检查一次签名证书直到收到或取消请求为止（类似 Ctrl+C）。

■注：可以使用 “--waitforcert” 选项修改 Puppet 客户端的等待时间。可以以秒为单位指定一个时间或设置为 0 使客户端不用再等待认证。

在主机上，需要签署证书。可以使用 puppetca 二进制命令签署。

```
puppet$ puppetca --list
gateway.example.com
```

■提示：Puppet 自带的二进制命令完整列表请参见 <http://reductivelabs.com/trac/puppet/wiki/PuppetExecutables>。

“--list” 选项列出所有等待签署的证书。可以使用 “--sign” 签署客户端使用的证书。

```
puppet$ puppetca --sign gateway.example.com
Signed gateway.example.com
```

■注：可以使用 “puppetca --sign --all” 命令签署所有等待的证书。

在客户端，签署证书后 2 分钟，可以看到如下条目。

```
notice: Got signed certificate
notice: Starting Puppet client version 0.24.7
err: Could not retrieve catalog: Could not find default node or by name with
'gateway.example.com, gateway' on node gateway.example.com
```

现在主机已认证客户端，但却还有一个出错信息。

```
err: Could not retrieve catalog: Could not find default node or by name with
'gateway.example.com, gateway' on node gateway.example.com
```

客户端已经连接，但是由于没有为客户端做任何配置，所以收到了此错误信息。

警告：主机和客户端的时间要准确一致，这一点很重要。SSL 连接依赖于矫正过的主机时钟。如果时钟不正确，则连接会出错，表明认证不可信。可以使用第 9 章中讨论过的 NTP 来确保主机的时钟正确。

19.2.5 创建第一个配置

现在主机已连接，下面就在其中添加一些配置。在 Puppet 主机上，需要添加一个节点定义和应用于客户端的配置。

先配置节点。在 `/etc/puppet/manifests/nodes/` 目录中创建一个名为 `gateway.example.com.pp` 的文件。此文件的内容如列表 19-10 所示。

列表 19-10 节点配置

```
node "gateway.example.com" {
    include sudo
}
```

`node` 指令为 Puppet 定义了一个节点或客户端配置。每一个客户端都需要一个 `node` 指令，以在其内部定义要应用到客户端的配置。指定客户端名称，并包括在双引号中，然后在花括号中指定应用到它的配置选项。

注：也可以指定名为 `default` 的特殊节点。若没有已定义的节点，则将此默认节点应用到客户端中。

可以在一个 `node` 指令中指定多个客户端，每个之间用逗号隔开。

```
node "gateway.example.com", "headoffice.example.com" {
    include sudo
}
```

注：此时，不能使用通配符（比如，`*.example.com`）来定义节点。然而，在 Puppet 中并没有继承模式使一个节点从其他节点中继承设置值。关于节点继承的内容参见 <http://reductivelabs.com/trac/puppet/wiki/LanguageTutorial#nodes>。

在节点定义中还有一个 `include` 指令。`include` 指令在客户端配置中添加了一个类（资源集合）。此处添加了一个名为 `sudo` 的类。可以使用多个 `include` 指令或使用逗号分隔每个类来引入多个类。


```
include sudo,sshd
```

下面将此类添加到 Puppet 配置中。在 `/etc/puppet/manifests/classes` 目录中创建一个名为 `sudo.pp` 的文件。列表 19-11 中为此文件的内容。

列表 19-11 sudo 类

```
class sudo {
  package { sudo:
    ensure => present,
  }

  file { ["/etc/sudoers":
    source => "puppet://$puppetserver/files/etc/sudoers",
    owner => "root",
    group => "root",
    mode => 0440,
  ]
}
```

版本控制

当配置变得越来越复杂时，应该考虑添加一个类似于 Subversion 的版本控制系统。版本控制系统允许用户记录和跟踪文件变化，此系统通常由软件开发人员使用。对于配置管理，版本控制使得用户可以跟踪配置的更改。若要恢复到先前的一致状态，或要更改配置而不影响主机的运行，这个工具是非常有用的。

关于如何使用 Subversion 的信息，请参考 <http://svnbook.red-bean.com/>，关于如何在 Puppet 中使用的特殊方法，请参考“<http://reductivelabs.com/trac/puppet/wiki/VersionControlPuppet>”。

上面添加了一个类指令并命名为 `sudo`。类的内容被包含在了花括号中。

在类中指定了两个资源，即一个 `package` 资源和一个 `file` 资源。软件包资源“`Package["sudo"]`”使用 `ensure` 属性规定必须要安装 `sudo` 软件包并将其值设置为 `present`。要移除此软件包，则要设置 `ensure` 属性的值为 `absent`。若要保证 `sudo` 软件包永远是最新的，则设置 `ensure` 属性的值为 `latest`，如下所示。

```
package { sudo:
  ensure => latest,
}
```

每次 Puppet 运行时，客户端都要检查目前安装的 `sudo` 软件包版本是否是最新的。如果是最新版本，Puppet 无任何动作；若还有更新的版本，Puppet 则将安装此版本。

Puppet 使用默认的软件包管理器来管理软件包。比如，在 Red Hat 中使用 `yum`，而在 Debian 中使用 `aptitude` 来安装、移除或升级软件包。这是 Puppet 中的方便功能之一——只需指定软件包资源，Puppet 就会自动查找适当的软件包管理器并使用它来安装所需的软件包。除此之外用户不用做任何事情，甚至不用了解软件包管理器是如何工作的。

■注：我们发现，Puppet 将其能配置的各种项目都称为类型（type），比如 `package` 类型。与特定操作系统相呼应的代码（比如，与 Yum 软件包管理器相对应的代码）称为 `provider`。每种类型可能有多个提供者。例如，`package` 类型就有针对 Yum、Aptitude、up2date、Ruby Gems、port、portage、rug 和 OSX DMG 文件以及很多其他软件包管理器的提供者。Package 提供者允许 Puppet 在各种各样的 Unix 操作系统和 Linux 发行版上配置软件包。

接下来,指定文件资源“File[“/etc/sudoers”]”。前面已经介绍过此资源的部分属性: owner, group 和 mode 属性。source 属性允许 Puppet 从 Puppet 文件服务器中获取文件并将其传递到客户端中。此属性的值是 Puppet 文件服务器的名称和位置以及要获取的文件名。

```
puppet://$puppetserver/files/etc/sudoers
```

将此值拆开来。 “puppet: //” 部分规定 Puppet 将会使用 Puppet 文件服务器协议来获取文件。

■注: 这是目前唯一可用的协议。在 Puppet 未来的版本中, 文件服务器将会支持其他协议, 比如 HTTP 或 rsync 等。可以期待此支持在 0.25.0 版本中出现。

“\$puppetserver” 变量中是 Puppet 服务器的主机名。创建此变量并放置在前述的 site.pp 文件中。除了使用此变量, 还可以直接在此指定文件服务器的主机名。

```
puppet://puppet.example.com/files/etc/sudoers
```

source 值的下一部分指定了文件共享和要发布的文件。这里共享的是前面在 fileserver.conf 文件中创建的 files, 要加载的文件是/etc/sudoers。假设文件 sudoers 在目录/etc/puppet/manifests/files/etc/sudoers 中。下面复制一个 sudoers 文件, 在主机中使用默认的 sudoers 文件。

```
puppet$ mkdir -p /etc/puppet/manifests/files/etc/
puppet$ cp /etc/sudoers /etc/puppet/manifests/files/etc/sudoers
```

创建一个 Puppet 配置

将现存的配置转化到 Puppet 上的最好办法是先从一个小的配置开始。选择一个功能模块或应用程序, 比如 sudo 或 SSH 守护进程, 并将其配置管理由手动转换为 Puppet 可管理的。当这些功能模块运行稳定后, 在添加其他的组件到 Puppet 配置中。完成此项任务的一个好方法是通过其功能来为主机分类。例如, 主机 gateway.example.com 运行了一系列服务, 如 Apache、Postfix 和 OpenVPN 等, 因此第一个逻辑步骤是配置这些服务, 然后再慢慢地将此主机所支持的其他功能模块添加进去。

19.2.6 应用第一个配置

创建了第一个配置文件后, 要把它应用到客户机中去。回到 gateway.example.com 主机, 在此运行 Puppet 客户端, 如列表 19-12 所示。

列表 19-12 应用第一个配置

```
gateway$ puppetd --server=puppet.example.com --no-daemonize --verbose
notice: Starting Puppet client version 0.24.7
info: Caching catalog at /var/lib/puppet/localconfig.yaml
notice: Starting catalog run
info: Filebucket[/var/lib/puppet/clientbucket]: ➡
Adding /etc/sudoers(a8ae43fcf346af54d473b13b17d6d037)
notice: //Node[gateway.example.com]/sudo/File[/etc/sudoers]: Filebucketed to with ➡
sum a8ae43fcf346af54d473b13b17d6d037
notice: //Node[gateway.example.com]/sudo/File[/etc/sudoers]/source: replacing from ➡
```



```
source puppet://puppet.example.com/files/sudoers with contents ➡
{md5}7255bc94cd66fc3416f991aed81ab447
notice: //Node[gateway.example.com]/sudo/File[/etc/sudoers]/mode: mode changed '640' ➡
to '440'
notice: Finished catalog run in 3.52 seconds
```

■ **提示：**Puppet 在 Red Hat 中记录日志到 `/var/log/messages` 文件中，在 Ubuntu 中，记录日志到 `/var/log/daemon.log` 中。

运行的第一步，输出了一个 `filebucket` 行。`filebucket` 是用来备份文件的特殊类型，读者应注意到我们并没有指定此类型。Puppet 自动备份将要修改或替换的文件（示例中，主机已经有一个 `/etc/sudoers` 文件并将使用从 Puppet 主机中来的新文件替代）。Puppet 将复制此文件到客户端中的一个目录中，通常在 `/var/lib/puppet/clientbucket` 目录下，这就意味着如果要将此文件取回，可以手动获取。

■ **提示：**Puppet 也可以使用 `filebucket` 类型将文件备份到主机中，参见“<http://reductivelabs.com/trac/puppet/wiki/TypeReference#filebucket>”。

文件备份后，Puppet 从主机上复制新的 `/etc/sudoers` 文件。

■ **提示：**Puppet 还有一个名为 `noop` 的测试模式。在此模式下，Puppet 不会真的升级主机配置，而是只告诉用户它将会做什么。这在应用配置之前对配置进行测试时是很有用的。可以使用带有“`--noop`”选项的 `puppetd` 命令使 Puppet 客户端运行在 `noop` 模式下。

最后，Puppet 将新文件的访问权限更改为 `0440`。但为什么 Puppet 没有修改此文件的所有者和用户组呢？这是因为在这里文件已经属于 `root` 用户组，因此 Puppet 不做任何改变。只有在客户端上有需要修改之处时，Puppet 才会加以修改。若当前配置无误，则 Puppet 无任何动作。

就是这样了。Puppet 已经配置好了客户端。如果 Puppet 客户端当前作为守护进程运行，它将等待 30 分钟（默认设置）后再次连接到主机，检查客户端配置是否已更改或在主机上是否存在新的配置。可以在配置文件 `/etc/puppet/puppet.conf` 中使用 `runinterval` 选项调整此运行时间间隔。

```
[puppetd]
runinterval=3600
```

这样就将时间间隔调整为 3600 秒，或 60 分钟。

Puppet 最佳用法

Puppet 配置可能会变得异常复杂。Puppet 用户之一——斯坦福大学写过一本最佳使用指导书，提供了一些有关如何配置 Puppet 的建议。Puppet 最佳用法指导书参见 <http://reductivelabs.com/trac/puppet/wiki/PuppetBestPractice>。切记，本文档仅包含主题，而且其内部信息可能不完全适合读者的环境。

19.2.7 为多个主机定义配置

前面仅仅触及了 Puppet 配置的表层功能，下面来看一下如何将当前的配置扩展到多个客

户端或节点中、如何区分两个客户端中的配置以及如何分别应用稍有不同的配置。

可以使用 Puppet 的伙伴工具 **Facter** 来进行这种差异性部署。**Facter** 是一个可以返回主机情况的系统查看工具。可以从命令行运行 **Facter** 并使用 **facter** 二进制命令来查看对主机 **gateway.example.com** 所知的信息。

```
gateway$ sudo facter
architecture => i386
domain => example.com
facterversion => 1.5.2
fqdn => gateway.example.com
hardwareisa => i686
hardwaremodel => i686
hostname => gateway
id => root
interfaces => eth0,eth1
ipaddress => 192.168.0.254
ipaddress_eth0 => 192.168.0.254
ipaddress_eth1 => 10.0.2.155
kernel => Linux
kernelrelease => 2.6.18-92.el5
kernelversion => 2.6.18
operatingsystem => RedHat
operatingsystemrelease => 5
...
```

上面显示了 **Facter** 所知主机情况的一个小集合，但可以看出它对主机知之甚多，包括主机名、网络信息、操作系统甚至是操作系统的发行版本号。

这些信息对 **Puppet** 有何用处呢？这些信息在 **Puppet** 中都是可用的变量。**Puppet** 在应用任何配置之前运行 **Facter**，收集客户端信息，然后将信息传送到 **Puppet** 主机，以用来配置客户端，例如，**hostname** 信息在 **Puppet** 配置中的变量为 **\$hostname**。列表 19-13 是一个信息使用示例。

Facter 更多信息

Facter 支持使用环境变量添加信息。客户机上任何带有前缀 **FACTER** 的环境变量（例如，**FACTER_LOCATION**）在 **Puppet** 中都有类似于 **\$location** 的相应变量。更多信息参见“<http://reductivelabs.com/trac/puppet/wiki/FrequentlyAskedQuestions#can-i-access-environmental-variables-with-facter>”。

Facter 还有高度的可扩展性。可以使用少量的 Ruby 代码添加用户自定义的信息，比如，用户环境自定义信息。关于如何添加自定义信息，请参考“<http://reductivelabs.com/trac/puppet/wiki/AddingFacts>”。

列表 19-13 使用 Fact

```
class sudo {
  package { sudo:
    ensure => present,
  }
  file { ["/etc/sudoers":
    source => "puppet://$puppetserver/files/$hostname/etc/sudoers",
    owner => "root",
```



```

    group => "root",
    mode => 0440,
  }
}

```

可以看到前面在“File[“/etc/sudoers”]”资源 source 属性中定义的稍加修改的 sudo 类。

```
puppet://$puppetserver/files/$hostname/etc/sudoers
```

在 source 属性的值中添加了 \$hostname 变量。现在 Puppet 不是从 /etc/puppet/manifests/files/etc/ 目录下查找文件，而是从 /etc/puppet/manifests/files/\$hostname/etc 目录中寻找。客户端连接后，\$hostname 变量将被所连接客户机的主机名代替，例如，若 gateway 主机连接，则 source 属性将变为 /etc/puppet/manifests/files/gateway/etc，这样就为特定客户机生成了一个不同的 sudoers 文件，如下所示。

```

/etc/puppet/manifests/files/gateway/etc/sudoers
/etc/puppet/manifests/files/headoffice/etc/sudoers

```

依据所连接的客户端，它们将各自获取一个适合自己的文件，但这并不是信息的唯一用途。还可以使用信息来确定如何配置特定节点，如列表 19-14 所示。

列表 19-14 一个 Fact 的 case 声明

```

node default {

  case $operatingsystem
    redhat: { include redhat } # include the redhat class
    ubuntu { include ubuntu } # include the ubuntu class
    default: { include generic } # include the generic class
  }
}

```

这里创建了默认节点定义，此配置将会应用到所有没有明确配置定义的节点。在此节点定义中，用到了 Puppet 语言的一个特性，即一个 case 声明。case 声明是一个对很多编程语言都通用的概念，它依据变量的值给出一个结果，在这里，\$operatingsystem 信息包含了在客户端中运行的操作系统名称（例如，Red Hat 值为 redhat，Ubuntu 值为 ubuntu）。

提示：Puppet 还有其他两种类型的条件语句，即 selector 和 if/else 语句。相关内容参见“<http://reductivelabs.com/trac/puppet/wiki/LanguageTutorial#conditionals>”。

列表 19-14 中，如果 \$operatingsystem 的值是 redhat，则 redhat 类将被加载到客户端。如果值是 ubuntu，则加载 ubuntu 类。最后一个值 default 是当值不匹配 redhat 或 ubuntu 中的任何一个时的动作。示例中若没有匹配到，则在客户端应用 generic 类。

在 case 声明中，也可以通过使用逗号分隔而指定多个值。

```

case $operatingsystem
  redhat,centos: { include redhat } # include the redhat class
  ubuntu,debian { include ubuntu } # include the ubuntu class
  default: { include generic } # include the generic class
}

```

这样，如果值是 redhat 或 centos，那么将会加载 redhat 类，如果值为 ubuntu 或 debian，则会加载 ubuntu 类。

列表 19-15 中是 Puppet 的另一种条件语句 selector。

列表 19-15 一个 Selector

```
service { "sshdaemon":
  name => $operatingsystem ? {
    redhat => "sshd",
    ubuntu => "ssh",
    default => "ssh",
  }
  ensure => running,
```

在列表 19-15 中，引入了一个新的类型 `service`，用来管理主机上的各种服务。前面已经用 `sshdaemon` 作为 `service` 资源的标题，但也使用了另外一个名为 `name` 的属性来指定将用来启动或停止客户端服务的名称。使用了一个称为 `selector` 的 Puppet 语言结构，结合“`$operatingsystem`”信息来指定 `name` 属性。这是因为在指定的每一个操作系统上，SSH 守护进程都叫一个不同的名字。例如，在 Red Hat 中 SSH 守护进程的 `init` 脚本名为 `sshd`，而在 Ubuntu 中却称为 `ssh`。

`name` 属性使用“`$operatingsystem`”信息来指定在每一个发行版中守护进程的名称。反过来，Puppet 使用此属性来确定启动或停止哪一个服务。因此，如果“`$operatingsystem`”信息的值为 `redhat`，则 `service` 资源将使用 `sshd` 名称来管理 SSH 守护进程。当其值即不是 `redhat` 又不是 `ubuntu` 时则使用 `default` 值。

最后，`ensure` 属性设置为 `running` 以确保服务已经启动。也可以将 `ensure` 属性的值设为 `stopped` 以确保服务不会启动。

■注：Puppet 语言有很多有用的特性。此语言的完整教程参见“<http://reductivelabs.com/trac/puppet/wiki/LanguageTutorial>”。

19.2.8 相关资源

Puppet 中的资源还有一个 `relationship` 的概念。例如，一个 `service` 资源可能会与安装它的软件包有关联。使用此概念，可以在安装新版本的软件包后触发服务的重启动作。使用此特性可以做很多有用的事情。考虑列表 19-16 中的示例。

列表 19-16 请求资源

```
class ssh {
  service { "sshdaemon":
    name => $operatingsystem ? {
      redhat => "sshd",
      ubuntu => "ssh",
      default => "ssh",
    },
    ensure => running,
    require => File["/etc/ssh/sshd_config"],
  }
  file { "/etc/ssh/sshd_config":
    path => "/etc/ssh/sshd_config",
    owner => root,
```



```

    group => root,
    mode => 644,
    source => "puppet://$puppetserver/files/etc/ssh/sshd_config",
    notify => Service[sshd_daemon],
  }
}

```

列表 19-16 显示了一个名为 `ssh` 的新类，其中包含了在列表 19-15 中创建的 `service` 资源，创建了一个 `file` 资源来管理 `/etc/ssh/sshd_config` 文件。除了在 `service` 资源中的 `require` 和 `file` 资源中的 `notify` 外，其他的属性都已经见过了。这两个不是普通的属性，它们称为 `metaparameters`。下面来了解一下每一个元参数以及其作用。

元参数 `require` 允许为一个或多个资源创建关联。在 `require` 元参数中指定的任何资源都要在此资源之前进行配置，所以，Puppet 将在 “`Service[sshd_daemon]`” 资源之前处理并配置 “`File[/etc/ssh/sshd_config]`” 资源。这种方法确保在启动 SSH 守护进程服务之前已经安装了适当的配置文件。也可以对 `package` 资源做同样的设置。

```

class httpd {
  package { "httpd":
    ensure => present,
  }
  service { "httpd":
    ensure => running,
    enabled => true,
    require => Package["httpd"],
  }
}

```

在这里，`package` 资源 “`Package[httpd]`” 必须在 “`Service[httpd]`” 可以启动之前安装。

■提示：上面还在 “`Service[http]`” 资源中添加了 `enabled` 属性。若此属性的值为 `true`，则此属性就设置了在主机重启时自动启动服务（与使用 `chkconfig` 或 `update-rc.d` 命令类似）。

列表 19-16 中还指定了另一个名为 `notify` 的元参数。此元参数添加到了 “`File[/etc/ssh/sshd_config]`” 资源中。`notify` 元参数提示了对一个资源的修改和更新。此例中，若 “`File[/etc/ssh/sshd_config]`” 资源被修改（例如，若更新了配置文件），则 Puppet 将通知 “`Service[sshd_daemon]`” 资源，触发其运行并因此而重启 SSH 守护进程服务。

■提示：可以创建的其他两个关联是 `subscribe` 和 `before`。可以在如下链接处看到这两个关联并能了解到其他有用的元参数，见 “<http://reductivelabs.com/trac/puppet/wiki/TypeReference#metaparameters>”。

19.2.9 使用模板

除了要从 Puppet 文件服务器中获取文件，还可以使用模板功能在这些文件中应用特定的值以配置服务或应用程序。Puppet 模板使用称为 ERB 的 Ruby 模板语言（参见 <http://www.ruby-doc.org/stdlib/libdoc/erb/rdoc/>）。使用方法很简单，如列表 19-17 所示。

列表 19-17 使用模板

```
file { "/etc/ssh/sshd_config":  
    path      => "/etc/ssh/sshd_config",  
    owner     => root,  
    group     => root,  
    mode      => 644,  
    content   => template("/etc/ssh/sshd_config.erb"),  
    notify    => Service[sshd],  
}
```

在列表 19-17 中，使用和前面创建的一样的“File[/etc/ssh/sshd_config]”资源，但使用 content 属性替换了 source 属性。使用 content 属性，而不是从 Puppet 文件服务器中获取的文件，文件的内容可以从此属性中得到填充。文件的内容可以定义到一个字符串中，如下所示。

```
content => "this is the content of a file",
```

或者，像列表 19-17 所示的，可以使用 Puppet 的一个名为 template 的特殊功能。定义一个模板文件来使用模板功能，Puppet 会用适当的值来填充在此模板中的任何 ERB 代码。

列表 19-18 sshd_config 模板

```
Port 22  
Protocol 2  
ListenAddress <%= ipaddress_eth0 %>  
SyslogFacility AUTHPRIV  
PermitRootLogin no  
PasswordAuthentication no  
ChallengeResponseAuthentication no  
GSSAPIAuthentication yes  
GSSAPICleanupCredentials yes  
UsePAM yes  
X11Forwarding yes  
Banner /etc/motd
```

在列表 19-18 中只使用了一个 ERB 来定义 SSH 守护进程的 ListenAddress 为“<%=ipaddress_eth0%>”。这里指定 Puppet 将“\$ipaddress_eth0”变量的值设置为 ListenAddress。反过来，ipaddress_eth0 主机信息就是变量的值，其中包含了 eth0 接口的 IP 地址。

现在当连接一个应用“File[/etc/ssh/sshd_config]”资源的客户端时，客户端上的 ipaddress_eth0 信息的值会加入到模板中，然后应用到客户端的/etc/ssh/sshd_config 文件中。

不仅仅是可以指定变量，在一个 ERB 模板中还可以呈现各种各样的功能，包括基本的 Ruby 表达式。如何使用模板的更多详细信息可以参见“<http://reductivelabs.com/trac/puppet/wiki/PuppetTemplating>”，在如下连接处有另一个典型模板的示例，见“<http://reductivelabs.com/trac/puppet/wiki/Recipes/ResolvConf>”。

Puppet 从由 templatedir 配置选项所定义的目录中寻找模板，此选项通常默认定义到/var/lib/puppet/templates。下面将使用以后的定义和配置覆盖此设置。在 Puppet 主机的 puppet.conf 配置文件中，添加如下内容。

```
[puppetmasterd]  
templatedir=/etc/puppet/manifests/templates
```

在列表 19-17 中指定的模板现在位于/etc/puppet/manifests/templates/etc/ssh/sshd_config 中。

Puppet 与自动化配置

Puppet 可以与自动化配置环境和启动服务器结合使用。可以在如下链接处找到关于如何结合 Puppet 和 Cobbler 以及 Ubuntu 中的 Preseed 的相关内容, 见“<http://reductivelabs.com/trac/puppet/wiki/BootstrappingWithPuppet>”。

19.2.10 定义

前面已经讨论了 Puppet 集合中的一个类型：一个类，还有另外一个类型：定义或 `define` 指令。`Definition` 用来设置在客户端中多次出现的配置示例。对定义的最好的理解就是，它是一个可以使用参数来调用的可重用的配置片段。

重用性也是在类和定义之间的主要不同。类包含了资源的单个实例，比如一个类可能包含一个定义了 `httpd` 软件包的软件包资源。此软件包将只在节点上存在一次，此后软件包将会使用类来安装、移除或管理。但有些配置会在客户机中存在多次，比如，`httpd` 服务器可能会定义了多个虚拟主机。用户可以创建一个定义来配置虚拟主机并传递合适的参数来配置各个主机。由于每一个设置都不同，每次设置一个定义 Puppet 都将会配置新的虚拟主机。

使用 `define` 指令可以创建一个定义，为定义指定一个标题并在括号中列出所有参数。接下来设置定义的内容并包含在花括号中。列表 19-19 包含了一个运行脚本，用来配置新虚拟主机的定义。

列表 19-19 定义

```
define new_vhost ( $ipaddress, $domain ) {
    exec { "/usr/sbin/create_vhost --vhost $title --ip $ipaddress --domain ➡
$domain":
    }
}

new_vhost { vhost1:
    ip => "192.0.2.155",
    domainname => "vhost1.example.com"
}
```

在列表 19-19 中，创建了一个名为 `new_vhost` 的定义，其中使用了“`$ipaddress`”和“`$domain`”两个参数。定义中，使用了 `exec` 资源类型（这是有一种资源类型，执行一个外部的脚本）。在定义在 `exec` 资源类型的脚本中指定了 3 个变量：“`$title`”，前面提到过的“`$ipaddress`”和“`$domain`”变量。

■注：“`$title`”变量存在于所有的资源中并包含了资源的标题。

下一行才是事实上的 `new_vhost` 定义。如此命名就像定义了一个资源类型，其中指定了定义的名称、标题，此处是 `vhost1`（也就是“`$title`”变量的值）。接下来指定了要传递给定义的变量，定义的格式和在资源中指定属性一样。

若使用此定义，将会在客户端中看到如下所示的日志内容。


```
notice://new_vhost[vhost1]/Exec[/usr/sbin/create_vhost --vhost vhost1 --ip➡  
192.0.2.155 --domain vhost1.example.com]/returns: executed successfully
```

■提示：读者也看到过一个用于 `recipe` 的定义，见 “<http://reductivelabs.com/trac/puppet/wiki/Recipes/ResolvConf>”，前面也曾经链接到此页面。

19.2.11 更多 Puppet

本章中只是轻描淡写地介绍了一下 Puppet，还有很多内容需要了解。在下面的小节中，将会介绍本章中还没有涉及的话题，读者可做进一步的探讨，以便更好地应用 Puppet。

1. 模块

读者已经见过两个资源集合了，分别是类和定义，Puppet 还有另外一个复杂的多的名为 `module` 的集合类型。可以将类、定义、模板、文件和资源集合全部整合进模块中。模块是可移植的配置集合。例如，一个模块可能会包含用来配置 Postfix 或 Apache 所需的所有资源。

关于如何使用模块可以参考 “<http://reductivelabs.com/trac/puppet/wiki/PuppetModules>”。在此页面中还有大量用户提供模块的链接。几乎可以肯定有人已经写过读者想要的配置服务或应用程序的模块，读者可以直接下载并重用这些模块而省去自己写的必要。

关于如何创建自己的模块以及模块如何组织的内容可以参见 “<http://reductivelabs.com/trac/puppet/wiki/ModuleOrganisation>”。

2. 功能

Puppet 还有一个功能集合。Function 是一些可以在 Puppet 主机上运行以执行某些动作的有用命令。前面已经见过两个功能了：一个是 `template`，其作用是创建一个模板配置文件；另一个是 `include`，作用为节点指定类。还有很多其他功能，其中包括能够调用外部命令并返回执行结果的 `generate` 功能，以及用于在主机上记录日志信息的 `notice` 功能，此功能对于测试配置是很有用的。

读者可以在 “<http://reductivelabs.com/trac/puppet/wiki/FunctionReference>” 处找到一个完整的功能列表，在 “<http://reductivelabs.com/trac/puppet/wiki/WritingYourOwnFunctions>” 还有关于如何创建自定义功能的相关文档。

3. 汇报

Puppet 有针对发生在节点或客户端上的事件进行汇报的功能。汇报现在已经是一个基本的功能了，读者可以在 “<http://reductivelabs.com/trac/puppet/wiki/ReportReference>” 看到当前的报告，也可以在 “<http://reductivelabs.com/trac/puppet/wiki/ReportsAndReporting>” 找到一些关于如何使用汇报功能的示例以及如何创建自定义汇报功能的文档。

4. 外部节点

可以想象，当运行环境中有了多个节点后，配置将会变得异常复杂。若人工定义所有节点及其配置变得非常麻烦时，可以考虑使用一个称为 `external node`（外部节点）的特性来进行更好的规划。外部节点允许用户将节点及其配置信息保存在外部的资源中。例如，可以将节点信息保存在一个数据库或一个 LDAP 目录中。更多关于外部节点和 LDAP 节点的内容，请分别参考“<http://reductivelabs.com/trac/puppet/wiki/ExternalNodes>”和“<http://reductivelabs.com/trac/puppet/wiki/LDAPNodes>”。

5. 环境

Puppet 最有用的特性之一便是其对环境概念的支持。环境允许用户为一个特定的环境制定一个配置，比如可能会为开发环境、测试环境、实际使用环境分别制定一个配置。Puppet 允许用户为每一个环境保留平行的配置设置并将其分别应用到不同的客户端中。

这对于创建各种各样的不同情景的配置来讲是一个很强大的机制，例如，创建一个“开发”➤“测试”➤“应用生命周期”来管理自定义设计的基础设置和应用程序。还可以使用环境来为站点或安全区域保留独立的配置设置，比如，为非军事区（DMZ）和内部网络保留独立配置。环境设置的更多信息，请参考“<http://reductivelabs.com/trac/puppet/wiki/UsingMultipleEnvironments>”。

6. 记录配置信息

系统管理员最容易忽略的一个工作就是记录信息，不但要写下来，还要保持信息的更新。Puppet 中有很多有用的工具可以帮助用户记录配置文件清单和模块。运行 `puppetdoc` 二进制可执行文件就可以利用 Puppet 扫描清单和配置并产生一个 HTML 格式的文档，也可以生成其他格式的文档。关于记录配置清单的内容，请参考“<http://reductivelabs.com/trac/puppet/wiki/PuppetManifestDocumentation>”。

扩展 Puppet

Puppet 默认使用一个称为 Webrick 的内网 Web 服务器。此 Web 服务器只支持数量很少的客户端，通常为 30 到 50 台。要使 Puppet 支持多于此数量的客户机，就需要替换当前的 Web 服务器，使用 Mongrel。关于 Puppet 扩展的内容，请参考“<http://reductivelabs.com/trac/puppet/wiki/PuppetScalability>”和“<http://reductivelabs.com/trac/puppet/wiki/UsingMongrel>”。

19.2.12 Puppet 排错

Puppet 拥有一个庞大、互助的社区以及非常详尽的文档。此外，本书的作者之一——James Turnbull 写过一本专门介绍 Puppet 的书《抽丝剥茧话 Puppet》（Pulling Strings with Puppet Apress,2008）。除了这本书，读者还可以参考 Puppet 的 wiki 页面，见“<http://reductivelabs.com/trac/puppet/wiki>”。其中包含了如下列出的有用信息。

- 配置参考 <http://reductivelabs.com/trac/puppet/wiki/TypeReference>。
- 类型参考 <http://reductivelabs.com/trac/puppet/wiki/TypeReference>。

- 报告参考 <http://reductivelabs.com/trac/puppet/wiki/ReportReference>。
- 功能参考 <http://reductivelabs.com/trac/puppet/wiki/FunctionReference>。

Wiki 中的其他有用资源还有以下几个。

- 语言教程: <http://reductivelabs.com/trac/puppet/wiki/LanguageTutorial>。
- 开始指南: <http://reductivelabs.com/trac/puppet/wiki/GettingStarted>。
- FAQ: <http://reductivelabs.com/trac/puppet/wiki/FrequentlyAskedQuestions>。

另外,还可以在“<http://reductivelabs.com/trac/puppet/wiki/GettingHelp>”处找到 Puppet 的邮件列表、#puppet IRC 频道以及包括售票系统 (ticketing system) 在内的其他各种资源。

19.3 小 结

本章中,介绍了一些简单的自动化配置工具,这些工具可以使主机的安装和配置过程变得异常快速和便捷。读者应该学到了如下内容。

- 配置一个网络启动的基础设置。
- 使用选定的操作系统自动启动主机。
- 安装一个选定的操作系统并自动回答安装过程中的各种问题。

本章还介绍了一个配置管理工具 Puppet,此工具将有助于对环境进行一致和准确的配置。读者应该学到了如下内容。

- 安装 Puppet。
- 配置 Puppet。
- 使用 Puppet 管理主机配置。
- 使用 Puppet 的高级特性。

在下一章中,将会介绍如何应用虚拟化技术和虚拟服务器来廉价高效地部署应用环境。

第 20 章 虚拟化



Sander Van Vugt 编写

如果要运行多个操作系统实例，可以为每一个实例购买一台服务器。在过去，这是通常的解决方法。而现在，一个更好的办法是买一个或几个功能强大的服务器并在这些服务器上应用虚拟化技术。虚拟化技术允许在一个硬件环境中运行几个操作系统实例。可以使用虚拟化技术优化服务器的利用率，甚至可以使用它在桌面环境中创建测试环境。本章中，读者将会了解到虚拟化技术提供了哪些功能以及如何在 Linux 主机上安装虚拟化服务。

20.1 虚拟化解决方案

有很多种虚拟化的方案可供选择。然而在 Linux 服务器环境中，只有几个是真正可以使用的，如下所示。

- VirtualBox。
- VMware。
- Xen。
- KVM。
- OpenVZ。

在后续章节中，会对上述虚拟化环境作一一介绍。

20.1.1 VirtualBox

VirtualBox 是安装在计算机中作为一个应用程序来使用的，它同时支持 Windows 和 Linux 操作系统。VirtualBox 是作为一个桌面虚拟化解决方案进行设计开发的，这就意味着，如果基于某种原因需要在一个测试机上开展工作而不需将这些测试机用于实际工作，VirtualBox 是一个最好的选择。VirtualBox 是一个开放源码的解决方案，可以免费安装和使用。本章稍后的内容将介绍如何在计算机中安装 VirtualBox 以及如何使用 VirtualBox 创建虚拟机。

20.1.2 VMware

在 19 世纪 90 年代初期，VMware 改造了虚拟化技术并将其引入到了基于 Intel 架构的计算机中。该公司开始于一个现在被称为 VMware 工作站的软件，此软件提供了一种可以在主机操作系统之上创建虚拟机的虚拟化高级解决方案。随后，又有几个产品被加入到了 VMware 的产品线。鉴于 VMware 的商业性质，本书不会对其安装和使用做介绍，但由于 VMware 在虚拟化市场上的重要地位，读者还是应该熟悉该公司提供的不同产品之间的区别。目前 VMware 主要有以下 3 个版本。

- **VMware 工作站版：**此版本对于培训和开发环境是一个绝佳的选择。使用 VMware 工作站，可以创建虚拟机，还可以将虚拟机的状态保存为一个快照。使用快照可以轻松及时地在任何时候将虚拟机恢复到原始状态（假设已经创建了此虚拟机原始状态的快照）。VMware 工作站版是一个供个人用户使用的商业产品。
- **VMware 服务器版：**起初，VMware 服务器版是作为服务器的低端虚拟化解决方案来设计开发的。和 VMware 工作站版一样，它也作为主机操作系统的一个应用程序来运行。VMware 服务器版和 VMware 工作站版的最主要区别在于其管理界面不同。VMware 服务器版的管理界面使管理员可以很轻松地管理几个正在运行的 VMware 服务器软件。VMware 服务器版可以从“<http://www.vmware.com>”处免费下载。
- **VMware ESX 版：**VMware ESX 版是 VMware 的一个重大突破，它是一个企业级的虚拟化解决方案，可以紧密集成在一个定制版本 Red Hat Linux 中作为综合管理程序运行。它为服务器提供高度定制和优化的虚拟化解决方案，但它也仍然是一个运行在一个小型主机操作系统上的综合管理程序，这就意味着由虚拟机产生的所有指令必须经过捕获和翻译才能被计算机的硬件处理。VMware 还为 ESX 版提供了一些附加产品（可单独购买），这些附加产品可以使 ESX 版在企业环境中的应用更加方便。比如，VMware 高可靠性（HA）可以确保当一个 VMware 主机出现故障时虚拟机可以迁移到另一台主机上去。另一个常见的解决方案是 VMotion，此产品可以不用停机将虚拟机从一个服务器移植到另一个服务器上。可以在 VMware 的网站“<http://www.vmware.com>”中找到关于 VMware ESX 的更多信息。

20.1.3 Xen

Xen 是在剑桥大学开发的，它走的是一条全新的虚拟化道路。Xen 的特性在于两种不同虚拟化模式之间的不同。众所周知，Xen 项目发起人发明的这个全新的虚拟化道路就是半虚拟化。在半虚拟化环境中，消除了主机和客户机的差异。

在一个半虚拟化的环境中，计算机开机时就启动了一个综合管理程序。在这个综合管理程序上，运行着 Domain 0（又称为 Dom0）机，这个特殊的机器中运行着一些和硬件访问有关的特殊任务。Dom0 可以确保没有两台虚拟机同时访问硬件。半虚拟化解决的主要问题是虚拟机可以直接访问硬件，而不是需要经过主机操作系统的翻译。然而，使用半虚拟化也有

一个条件，那就是虚拟机必须使用一个经过调整的内核，这就意味着不是任何操作系统都可以用于半虚拟化的。比如，老版本的 Windows 系统就不能使用半虚拟化所带来的便利。

除了半虚拟化，Xen 还提供完全虚拟化模式。在完全虚拟化模式下，Dom0 的功能是作为一个操作系统捕获虚拟机产生的指令并将其翻译到综合管理程序层。为了尽量减少这种模式对系统性能的影响，完全虚拟化模式需要 CPU 自身虚拟化特性的支持。这种支持表现为 Intel CPU 上的 VMX 特性。

可以通过查看文件/proc/cpuinfo 的内容检查 CPU 是否具有此种支持。列表 20-1 中显示的是一个带有虚拟化支持的服务器中的此文件内容，可以看到 VMX 特性出现在了 flag 中。

列表 20-1 检查/proc/cpuinfo 文件确定 CPU 是否支持虚拟化

```
processor           : 3
vendor_id           : GenuineIntel
cpu family         : 6
model              : 15
model name         : Intel(R) Xeon(R) CPU           X3230 @ 2.66GHz
stepping           : 11
cpu MHz            : 2666.844
cache size         : 4096 KB
physical id        : 0
siblings           : 1
core id            : 3
cpu cores          : 1
fpu                : yes
fpu_exception      : yes
cpuid level        : 10
wp                 : yes
flags               : fpu de tsc msr pae mce cx8 apic sep mtrr mca cmov pat pse36
                   : clflush dts acpi mmx fxsr sse sse2 ss ht tm syscall nx lm constant_tsc pni monitor
                   : ds_cpl vmx est tm2 cx16 xtpr lahf_lm
bogomips           : 5335.56
clflush size       : 64
cache_alignment    : 64
address sizes      : 36 bits physical, 48 bits virtual
power management:
```

不论如何优化，使用完全虚拟化模式时，一定会有部分性能损失。为了弥补这方面的性能损失，一些厂商提供了商业的驱动程序包。比如 Novell 公司提供了 Novell 虚拟化驱动程序包，其中包含了一些驱动程序，使得虚拟机中的存储器和网络适配器可以在半虚拟化模式下使用。因为这些是使用最多的适配器，所以可以在一个完全虚拟化的环境中极大地受益于这种解决方案所带来的性能优化。图 20-1 给出了 Xen 虚拟化解决方案的示意图。

关于 Xen 的归属问题，可能有些混乱。最初，Xen 在剑桥大学创建（参见“<http://www.cl.cam.ac.uk/research/srg/netos/xen/>”）。Xen 的创立者把它作为一个开源的项目，因此其他硬件或软件公司也可以在 Xen 项目中提供自己的内容。Xen 的创立者也成立了名为 XenSource 的公司。

2007 年，Citrix 公司收购了 XenSource。基于 XenSource 公司的技术，Citrix 现在提供 Xen

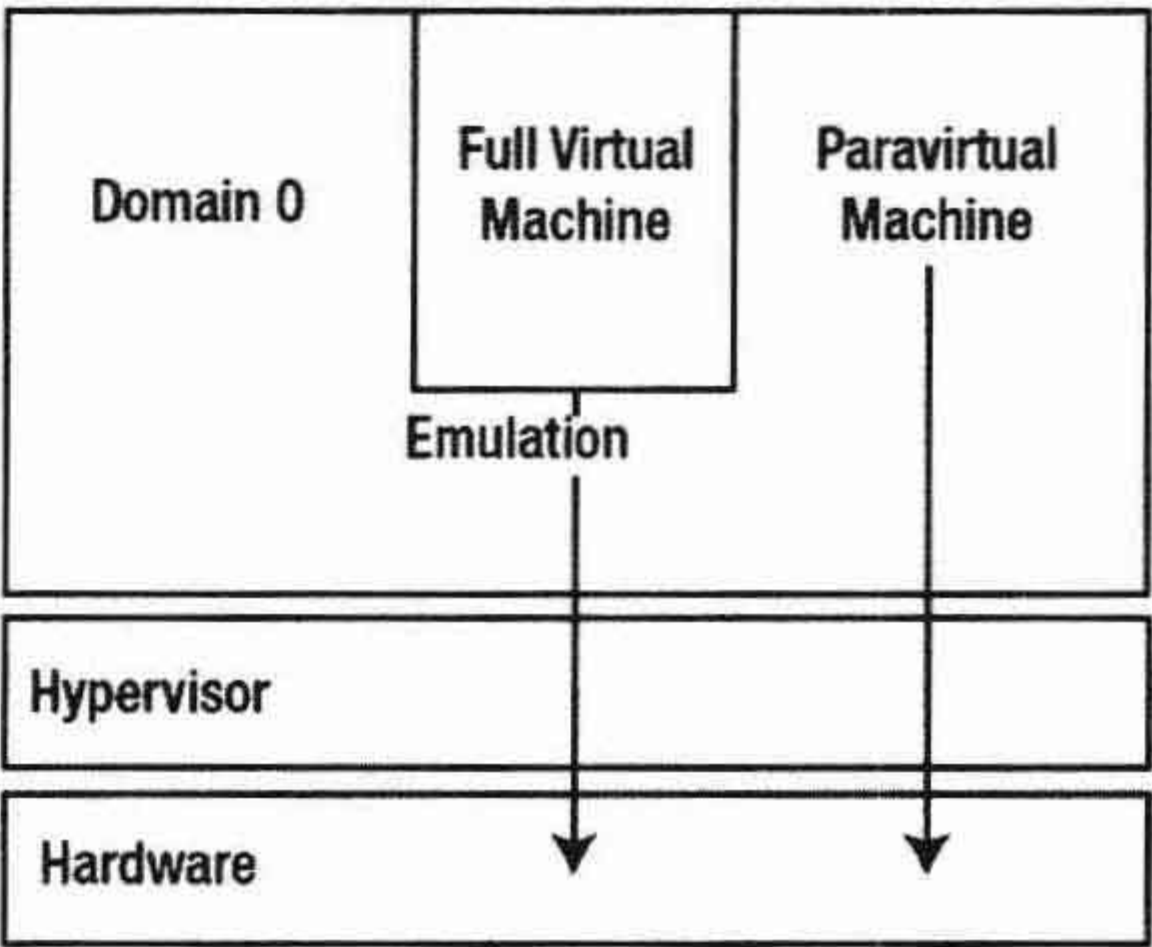


图 20-1 Xen 虚拟化

Server，这是一个带有 Citrix 公司独特管理工具的 Citrix 牌产品。除了这个产品，还有其他几个 Linux 版本的 Xen。实际上，有些 Linux 提供商已经不再使用 Xen，而是使用 KVM（参见 20.1.4 节），但 Novell 公司还是使用 Xen 作为其产品的默认虚拟栈。因此，简而言之，Xen 的创立者现在为 Citrix 工作，但 Xen 综合管理程序仍然是、而且将会一直都是一个开源的项目。

20.1.4 KVM

最后一个与 Linux 环境高度相关的虚拟化解决方案是 KVM。KVM 的工作方式与 Xen 很相似，但是内部却有重大的区别。Xen 是一个复杂的产品，它使用自带的管理程序，而 KVM 却只有一个加载到当前 Linux 内核中的内核模块。这种方式的主要优点就是使虚拟化更加容易实现。此时不再需要具体的管理程序，同时，实现虚拟化的只是一个内核模块而已，这就解决了 Xen 的重要缺陷：复杂。由于在 KVM 中只需处理一个内核模块，KVM 就使虚拟化不再那么复杂，KVM 在 Red Hat 和 Ubuntu 中都是默认的虚拟化解决方案。

20.1.5 OpenVZ

OpenVZ 是一个使用基于容器方法的虚拟化解决方案，这就意味着主机操作系统会启动一个通用的内核，然后在主机操作系统中使用容器来实现虚拟化。可以比较一下容器环境和 chroot 环境，在 chroot 环境中，应用程序也是在自己的上下文中运行。在 OpenVZ 中，应用程序就不再是一个应用程序了，而是一个在应用程序上下文中运行的完全的虚拟机了。因为虚拟机需要和主机操作系统直接通信，所以在 OpenVZ 环境中只能安装 Linux 虚拟机。

20.2 使用 VirtualBox

如果你是一位顾问、一位培训师或者一位开发人员，使用虚拟机的主要目的是进行测试，那么 VirtualBox 可能是最佳的选择。VirtualBox 作为一个应用程序可以安装在所有主要 Linux 发行版中，还可以在其中创建虚拟机。它的性能良好，尽管性能并不是创建 VirtualBox 时的主要设计目标。本章将会讨论如何安装 VirtualBox，以及如何在 VirtualBox 中创建虚拟机。

20.2.1 安装 VirtualBox

■注：VirtualBox 并不是需要专门安装在某个服务器中的应用程序。它也可以被安装到 Linux 桌面上，然后再在其中创建一个虚拟机来给客户展示 Linux 服务器。因此，本部分不会包含如何将其安装到 Ubuntu 服务器或 Red Hat 企业版 Linux 中的具体命令。相反，本部分会给出一般性命令，以帮助读者把它安装在几乎所有 Linux 发行版中。

多数发行版的默认软件库中并不包含 VirtualBox。因此，最好的方法是自己下载并安装软件包。下面的过程介绍了如何下载和安装 VirtualBox。为了安装更加简便，开始之前最好确认以下软件包是否已安装。

- Qt 4.3 或更新的版本。
- SDL 1.2.7 或更新的版本（查看 libSDL）。
- 内核源文件。

1. 大部分发行版从其软件库中就可以知道 VirtualBox 的开源版本。可以使用软件包管理器来安装。安装时使用如下命令。

```
yum install virtualbox-ose
```

如果在 Ubuntu 中安装 VirtualBox，使用如下命令。

```
apt-get install virtualbox-ose
```

2. 软件安装完成后，将用户账号加入到 vboxusers 用户组中。通常此用户组的 GID 为 1000，但最好还是检查文件/etc/group 来确认一下。下面的命令假定 vboxusers 的 GID 为 1000，并把用户 linda 加入到此用户组中。

```
groupmod -A linda vboxusers
```

3. 要启用 USB 支持，需要修改文件/etc/fstab 中加载 USB 文件系统的相应行内容。此行的默认内容如下所示。

```
usbfs          /proc/bus/usb  usbfs          noauto          0 0
```

将其修改为：

```
usbfs          /proc/bus/usb  usbfs          mode=0644,gid=1000 0 0
```

4. 到此为止，可以准备好重启系统，以激活所有修改过的配置了。

20.2.2 使用 VirtualBox 创建虚拟机

重启系统后，启动 Virtualbox。尽管安装时已经在图形化桌面上放置了一个图标，但第一次启动时最好还是使用当前用户账号打开一个控制台窗口并输入 VirtualBox 命令来启动它。这样在应用程序启动可以随时看到出错信息（如果有）。

如果一切正常，将会出现一个 VirtualBox 的注册窗体，如图 20-2 所示。在此窗体中需要提供用户名和 e-mail 地址进行注册，但是如果不希望在收件夹中收到来自 VirtualBox 的邮件的话，不要忘了选中“请不要使用此信息联系我（Please do not use this information to contact me.）”选项。

注册完成后，将会看到 VirtualBox 的主窗口。现在就可以使用如下过程安装一个虚拟操作系统了。

■提示：使用虚拟操作系统时，最好将计算机本地硬件驱动器中的安装媒体制作成 ISO 文件。可以使用 dd 命令完成此项工作。例如，下述命令将实际位于/dev/cdrom 的 CD 中的内容复制成一个名为/winxpsp.iso 的 ISO 文件，即“dd if=/dev/cdrom of=/winxpsp2.iso”。

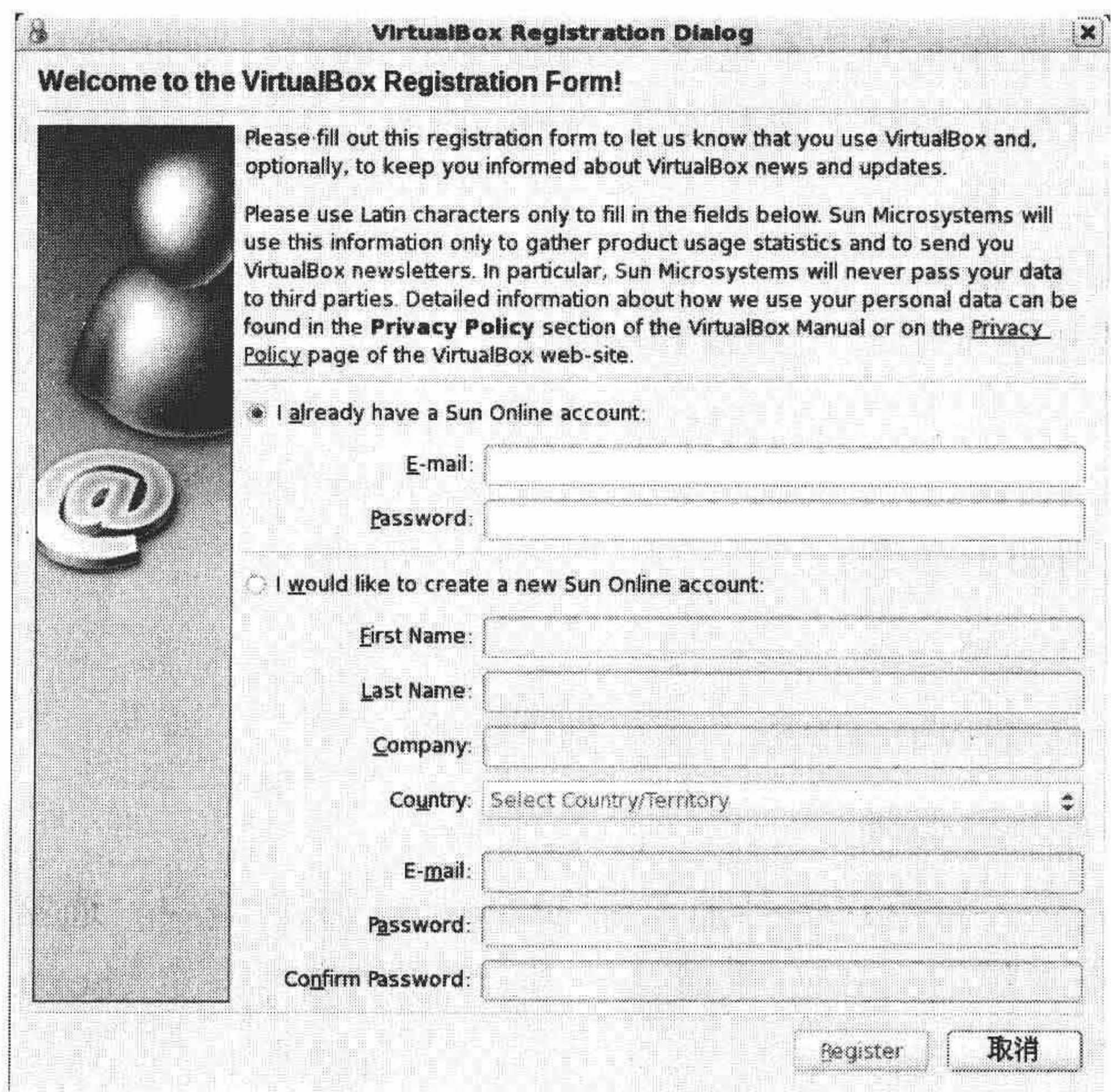


图 20-2 使用 VirtualBox 前，需要先注册

- 1. 在 VirtualBox 主窗口中单击“新建”。程序将会打开一个帮助用户完成创建虚拟机过程的向导窗体。在向导窗体中单击“下一步”。
- 2. 在窗口中为新建的虚拟机输入一个名字，同时选择要在虚拟机中使用的操作系统类型，然后单击“下一步”继续（如图 20-3 所示）。
- 3. 指定虚拟机要使用的 RAM 数量，单击“下一步”。向导会自动检测计算机中的内存总数，以免用户给虚拟机指定大于本机内存总数的数量（如图 20-4 所示）。

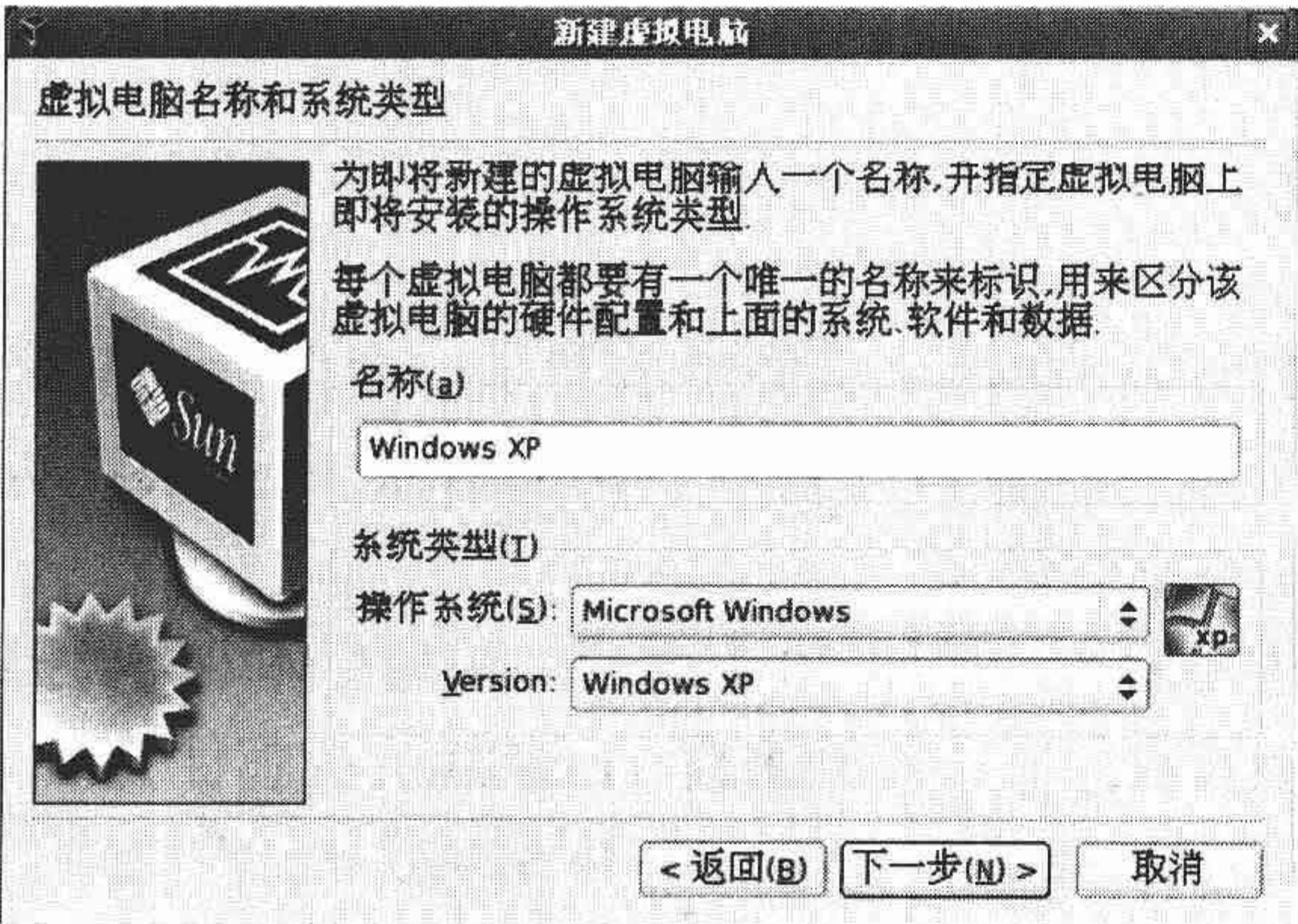


图 20-3 为虚拟机命名并选择操作系统类型

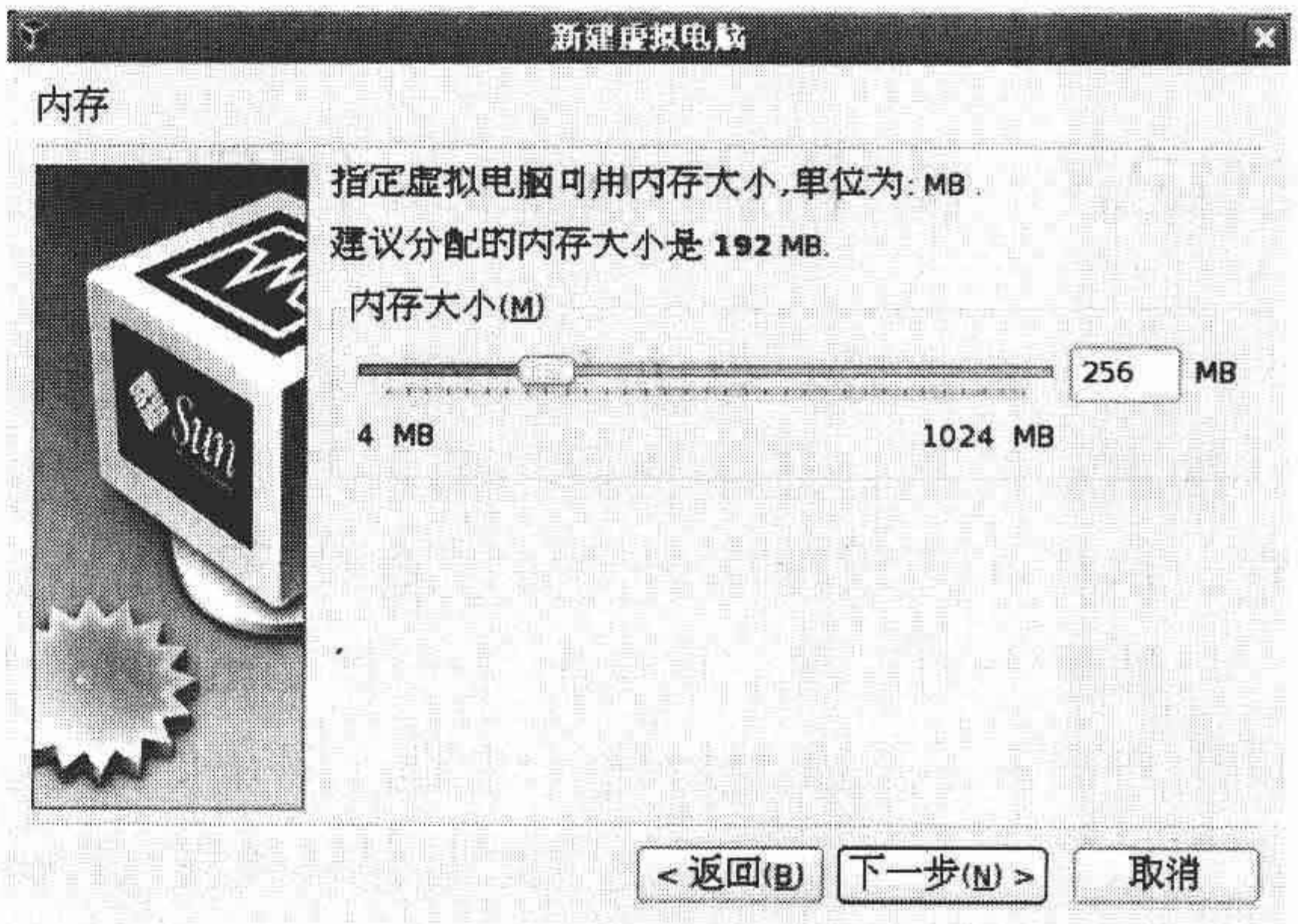


图 20-4 使用滑动条为虚拟机指定内存数量

- 4. 接下来，安装程序需要知道要分配给虚拟机的硬盘空间。如果此时还没有虚拟硬盘，可以单击“新建”按钮打开一个创建虚拟硬盘的向导。在此向导中，单击“下一步”开始创建。向导将会询问要创建何种类型的硬盘。如果读者只是想试用 VirtualBox，可以选择“动态扩展”，此选项将使用最少的硬盘空间，同时也运行很快。如果读者要创建一个虚拟机并确

定要长时间使用，可以选择“固定大小”。由于不会被分割，固定大小的镜像会更快，但创建的时间会更长一些。在图 20-5 中，选择了“动态扩展”。

5. 现在，需要为镜像文件命名并输入硬盘大小。如果只是为了试用，默认的 10GB 应该就足够了。如果不是，使用滑动条增加虚拟硬盘的大小并单击“下一步”。在下一步，也就是向导的最后一个页面中，将会列出上述所有选择的概览。在此页面中，单击“结束”。然后将返回到主向导页面，在此页面中选中了刚刚创建的虚拟硬盘文件。然后，单击“下一步”继续（如图 20-6 所示）。

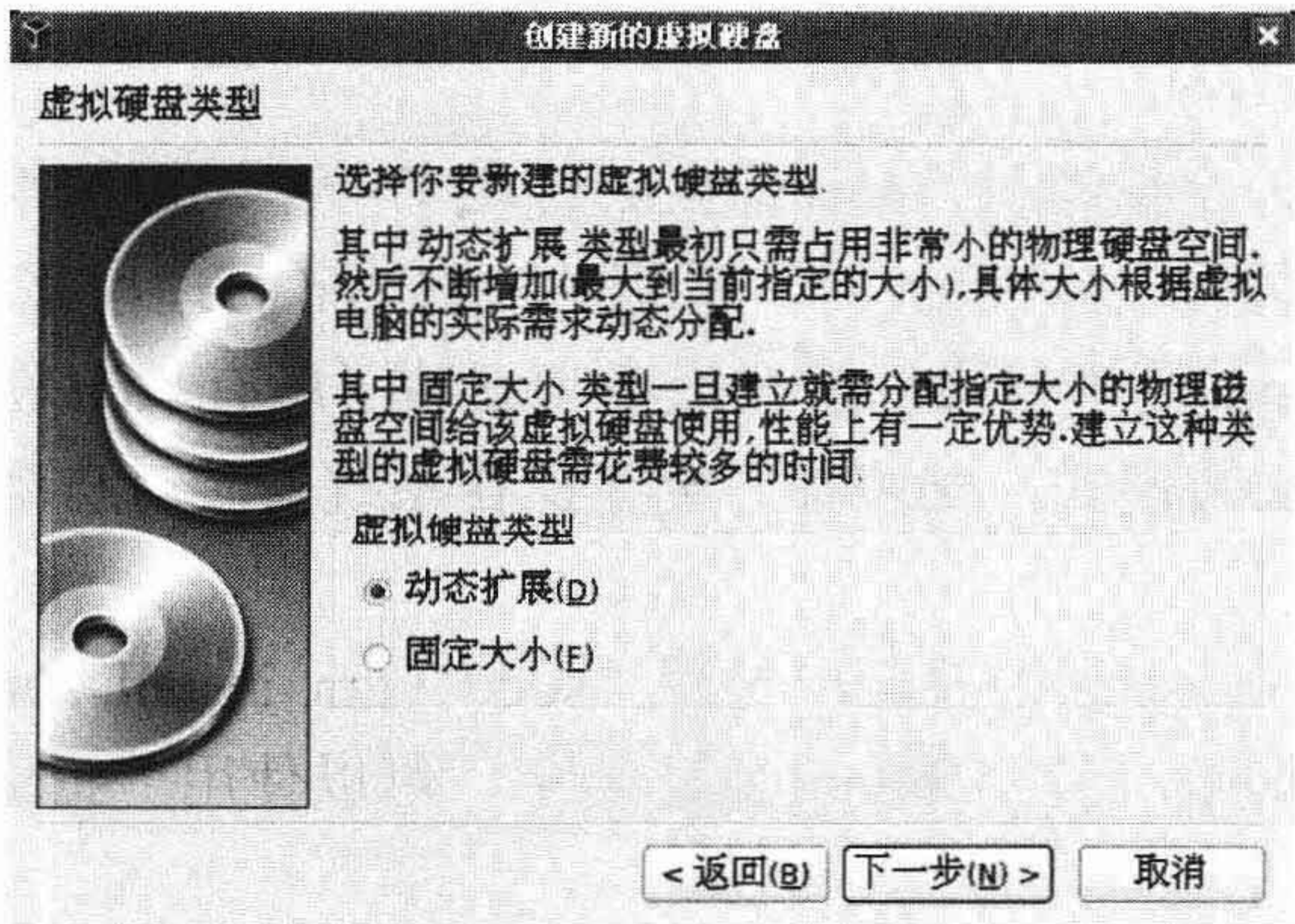


图 20-5 为了快速创建，选择“动态扩展”选项



图 20-6 回到主向导页面，单击“下一步”继续

6. 在创建虚拟机向导的最后一个窗口，单击“结束”回到 VirtualBox 的主界面。在这里会看到刚创建的虚拟机，已经基本上可以使用了。在这里，还需要指定要使用的光盘驱动器。特别是如果要安装一个新的操作系统，需要指定使用哪一个 CD 或 DVD（如图 20-7 所示）。

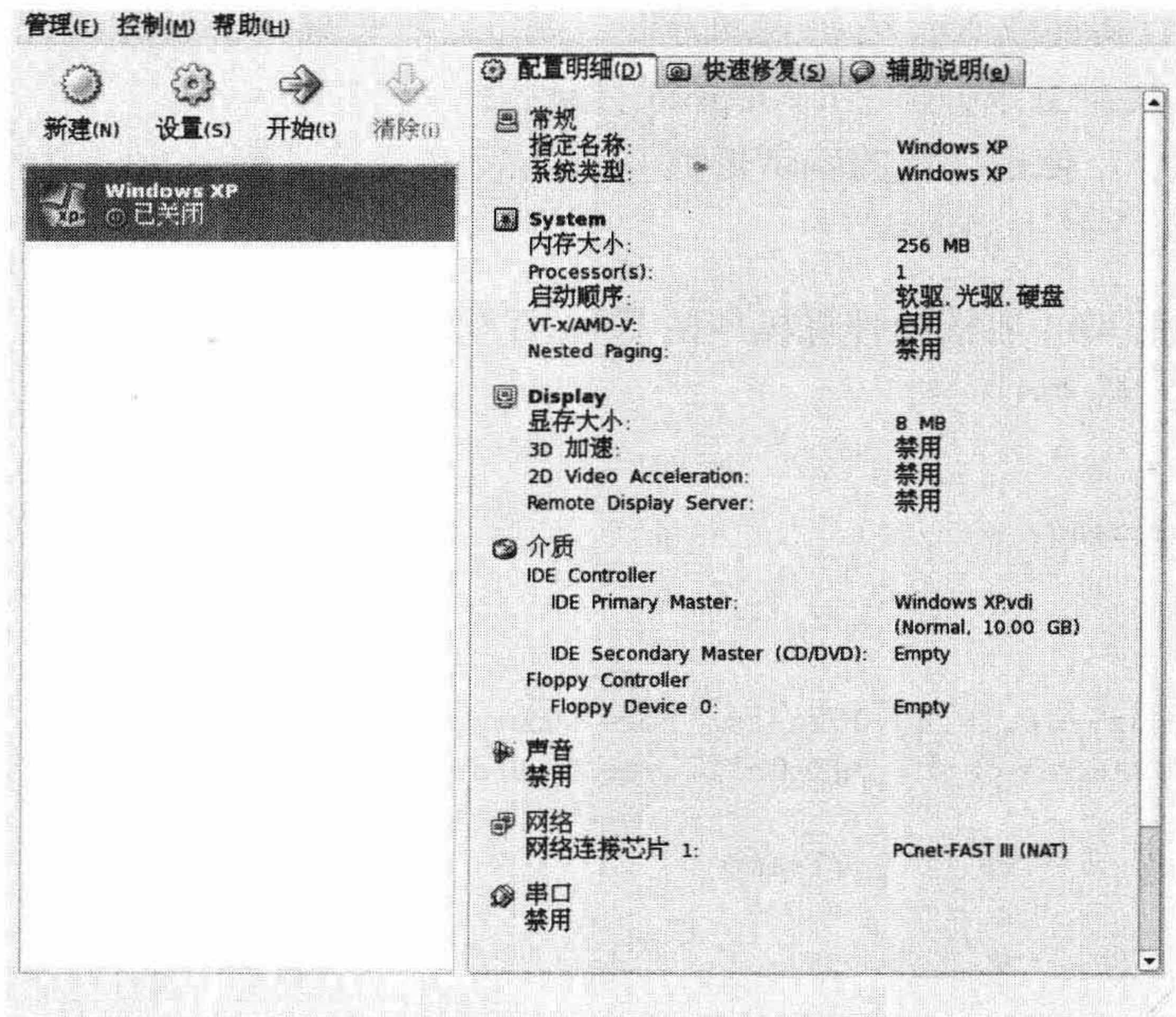


图 20-7 指定使用哪一个 CD 或 DVD

7. 现在单击 CD/DVD-ROM 链接并选中“挂载 CD/DVD 驱动器”选项。接下来，选择“ISO 镜像文件”并单击“浏览”按钮。系统将打开一个窗口使用户可以浏览文件系统并查找要使用的 ISO 文件。在 VirtualBox 主窗口中选择了文件之后，单击“开始”来启动虚拟机。假设已经选定了要安装的操作系统的安装媒体，此时将会开始虚拟操作系统的安装过程，接下来就可以和通常在真正的主机上一样安装操作系统了。

20.3 使用 Xen 安装虚拟机

在前面的章节中，使用 VirtualBox 安装了一个虚拟机。VirtualBox 提供了一个不需要用户启动特殊内核的易用方案。在 Xen 中却完全不同，使用 Xen 时，还需要一个 Xen 管理程序。本节中，将会介绍如何准备在计算机中使用 Xen。这里假设读者已经手动安装了所有 Xen 软件包。在 Red Hat 企业版中，如果选择了虚拟安装模式，安装程序会在操作系统安装时自动完成此项工作。

若要在安装完成的服务器中安装 Xen，在 Red Hat 中就可以使用“`sudo yum install xen kernel-xen virt-manager`”命令来安装所需的软件包，而在 Ubuntu 服务器中，可以使用“`sudo aptitude install ubuntu-xen-server`”命令。

20.3.1 为使用 Xen 准备计算机环境

使用 Xen 需要一个特殊的内核。如果已经在发行版中安装了 Xen 的所有软件包，应该确保计算机启动时是运行的这个特殊的 Xen 内核。当然，也可以在计算机启动时手动选择，但还是配置 GRUB 自动加载 Xen 内核更加方便。如下过程叙述了如何完成此项配置。

- 1. 使用 root 用户登录并激活 `/boot/grub` 目录。
- 2. 在此目录中，使用喜欢的编辑器打开 `grub.conf` 文件。列表 20-2 中给出了安装完 Xen 内核的计算机上此文件中的内容。

列表 20-2 安装完 Xen 内核的计算机中的 GRUB 配置文件

```
HKG:/boot/grub # cat menu.lst
default 1
timeout 8
gfxmenu (hd0,0)/message

title normal kernel
    root (hd0,0)
    kernel /vmlinuz-2.6.16.60-0.21-smp root=/dev/disk/by-id/scsi\
    -35000c5000ebacfc3-part3 vga=0x317 resume=/dev/sda2 splash=silent\
    showopts
    initrd /initrd-2.6.16.60-0.21-smp

title XEN
    root (hd0,0)
    kernel /xen.gz
```



```
module /vmlinuz-2.6.16.60-0.21-xen root=/dev/disk/by-id/scsi\
-35000c5000ebacfc3-part3 vga=0x317 resume=/dev/sda2 splash=silent\
showopts
module /initrd-2.6.16.60-0.21-xen
```

可以看到，列表 20-2 中的示例配置文件有两个部分。应该确保选定第二部分默认启动。可以使用 GRUB 配置文件中主要部分的 default 1 行来指定。

3. 重启服务器启动 Xen 内核。

20.3.2 创建 Xen 虚拟机

在一个 Xen 环境中，可以使用一个非常方便的工具虚拟机管理器来创建和管理虚拟机。除此之外，还可以使用一些命令行工具。由于从命令行创建虚拟机很麻烦，因此读者肯定希望在一个图形化用户环境中使用虚拟机管理器来创建虚拟机。下面的步骤讲述了如何完成此项工作。

1. 运行命令 virt-manger 启动虚拟机管理器。图 20-8 显示了虚拟机管理器的界面。

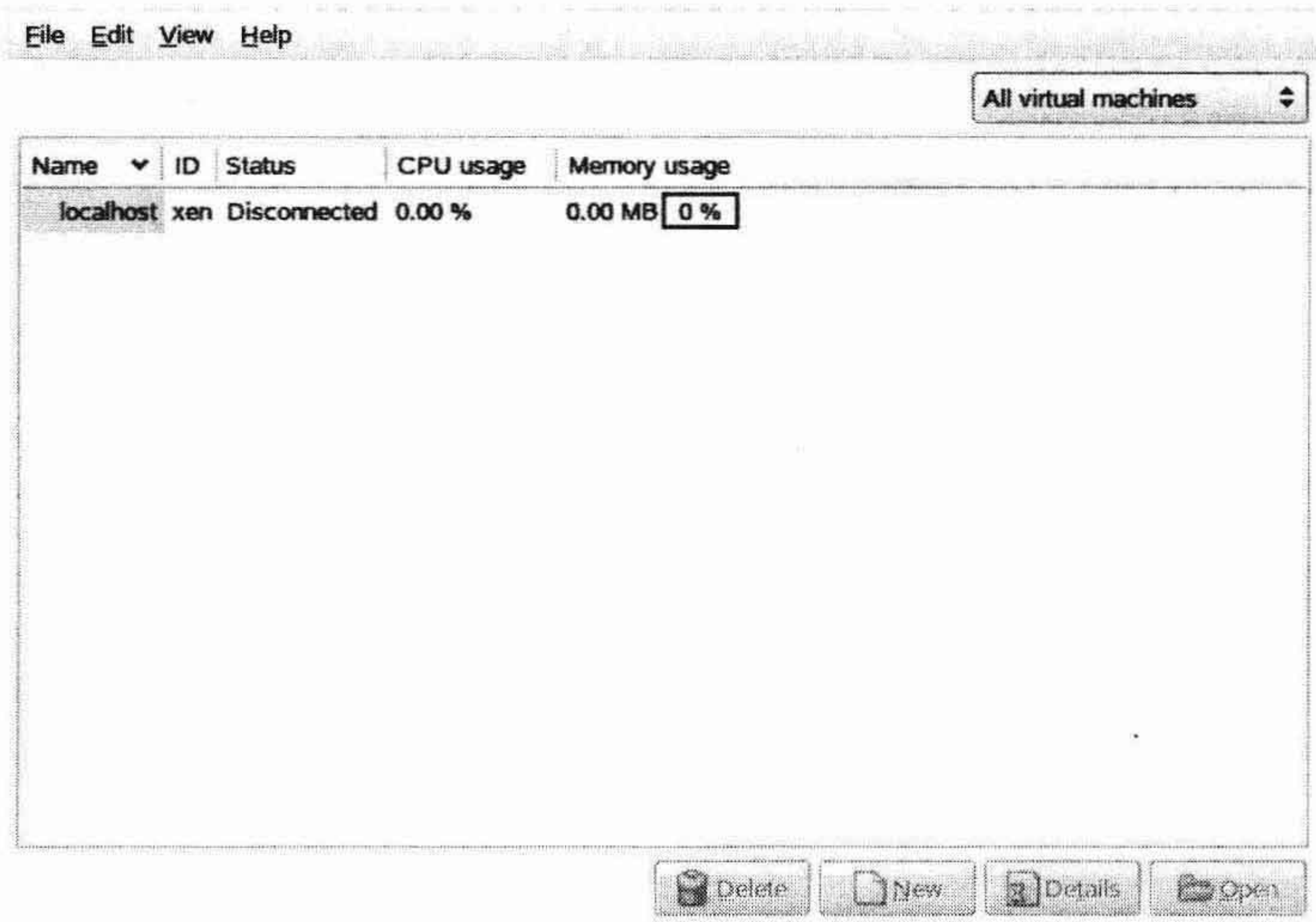


图 20-8 虚拟机管理器界面

- 2. 双击名为“localhost xen”的行，界面将列出计算机中现存的所有虚拟机。双击“localhost”后，可以单击“New”开始创建一个新的虚拟机（如图 20-9 所示）。
- 3. 单击“New”开始创建一个新的虚拟机。此时会启动一个小型的向导，在向导中需要填写创建虚拟机所需的所有参数。在向导页面上单击“Forward”。
- 4. 向导要求填写要创建的系统的名称。此名称随意，只要以后能识别出此系统是做什么用的就可以了。要确保在输入的系统名称中没有空格。
- 5. 现在要选择虚拟机的类型。若操作系统支持，最好选择“Paravirtualized”（如图 20-10 所示）。如果客户操作系统不支持（Windows 操作系统就这样），就只能选“Fully Virtualized”。为了能够安装完全虚拟化的虚拟机，要确认在计算机的 BIOS 中开启了虚拟化特性。
- 6. 需要指定 Xen 在何处可以找到安装文件。默认不支持从本地媒体安装，可以输入包含安装文件库的安装服务器地址（如图 20-11 所示）。

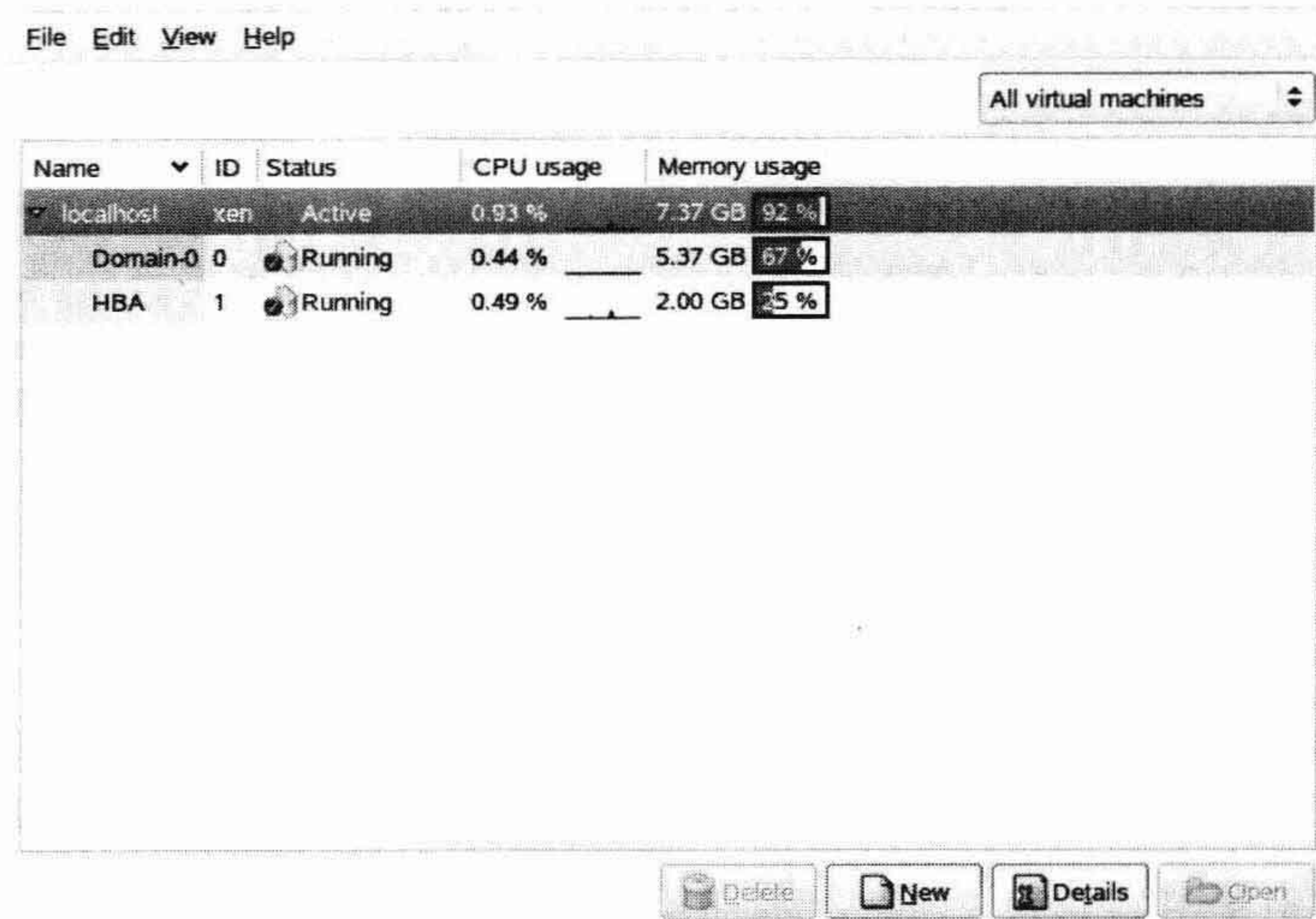


图 20-9 双击“localhost”行添加一个新的虚拟机

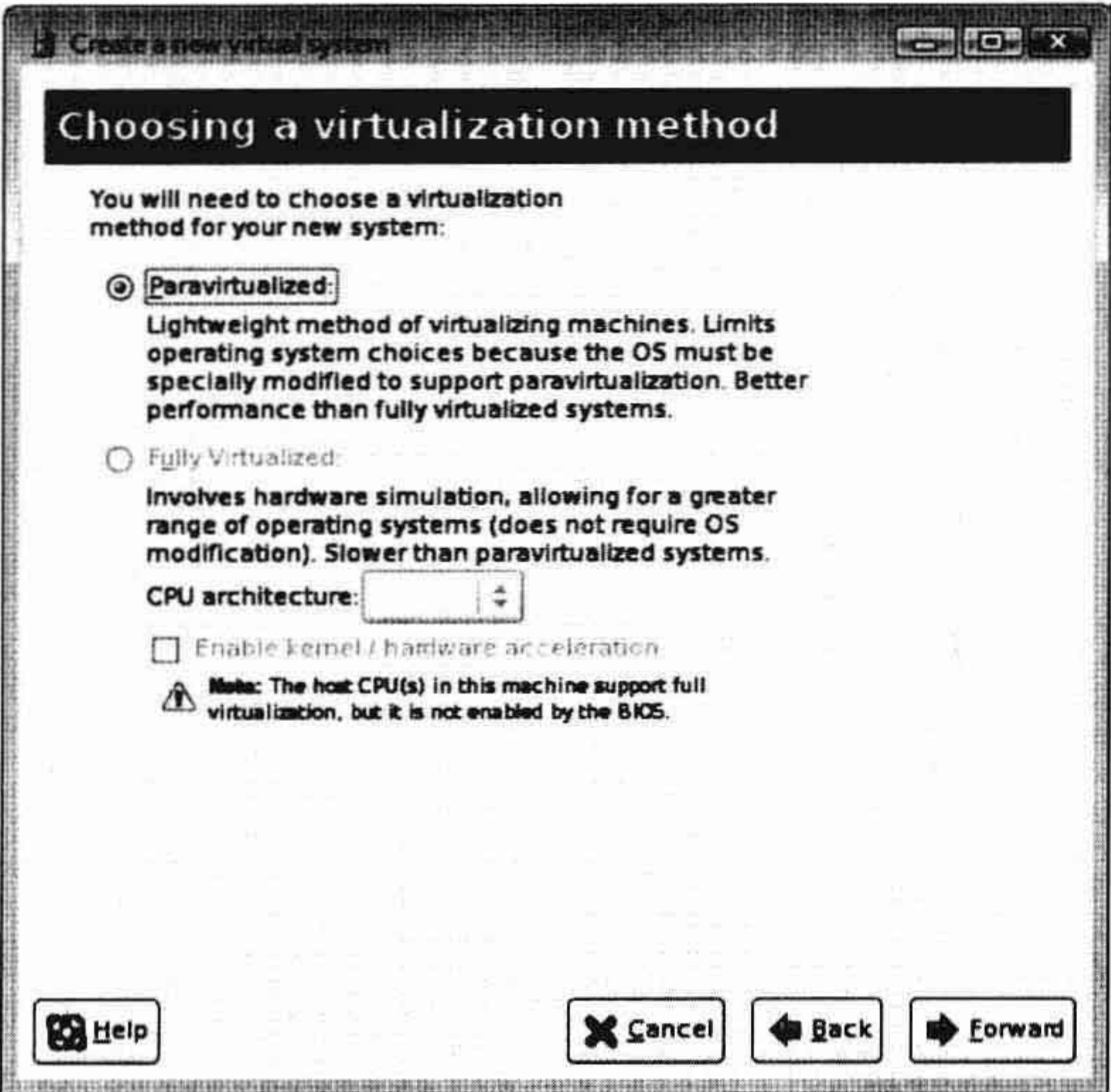


图 20-10 如果操作系统支持，半虚拟化是最好的选择

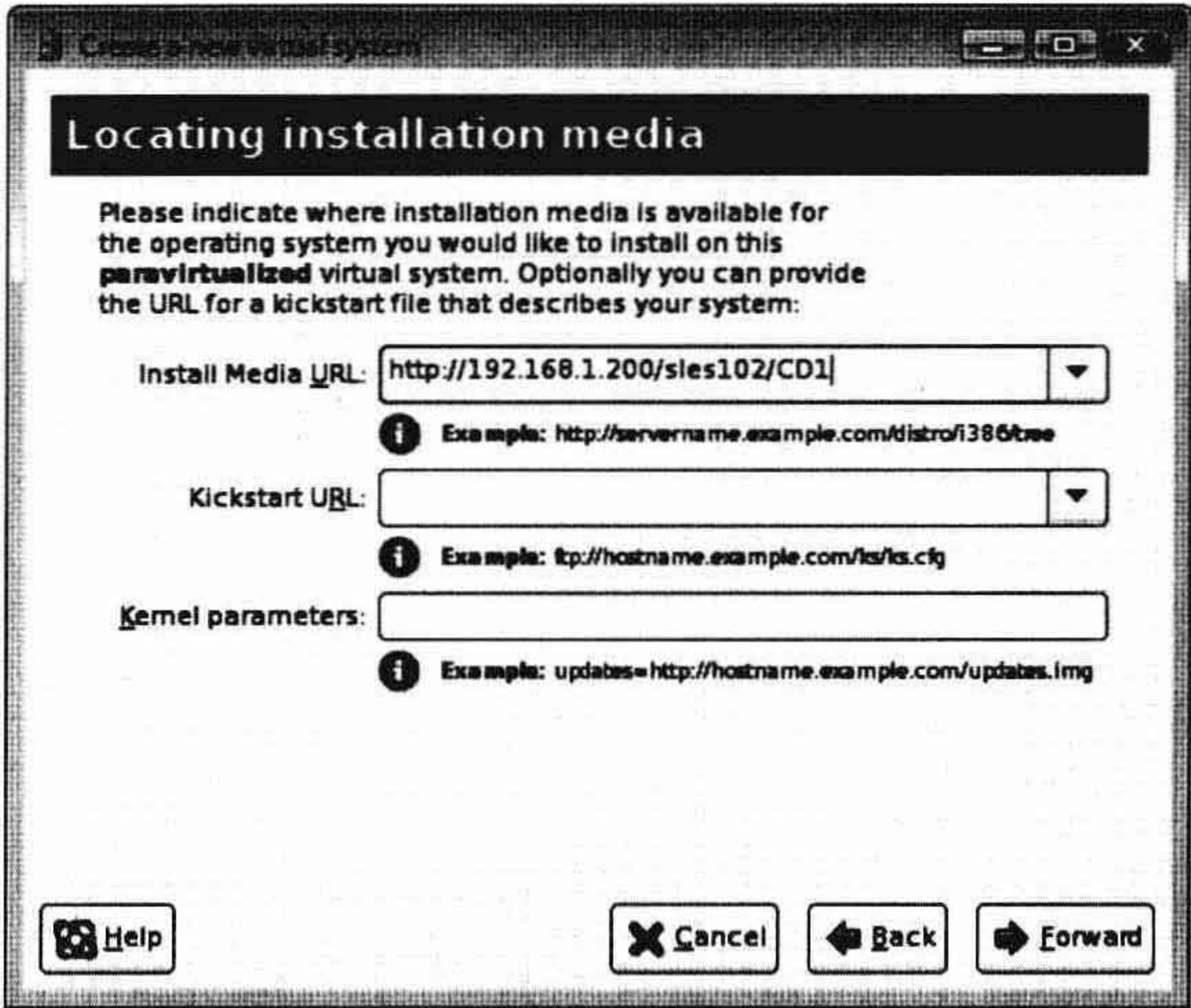


图 20-11 输入安装源文件的地址

7. 还要给分配到虚拟机的虚拟硬盘指定想要使用的存储终端（如图 20-12 所示）。如果要创建一个简单的安装，可以使用一个磁盘文件作为存储终端。但是为了获得更好的性能，还是建议将虚拟磁盘定位于客户机的一个磁盘分区或一个逻辑卷。为了简单起见，本章将演示如何使用文件作为存储终端来安装虚拟机。文件的默认大小为 4000MB，这对于一个小型测试机来讲应该足够了。然而如果读者在使用的过程中想增加此文件大小，最好选中选项“Allocate entire virtual disk now?”，因为这样可以确保文件使用的是一个连续的磁盘空间，从而避免出现磁盘碎片。

8. 接下来，需要配置网络。默认的选项是使虚拟机工作在只与主机共享的虚拟网络中。在实际使用中这并不是一个好的选择，这里将其改为选择“Shared Physical device”选项，网络相关的其他内容保持默认（见图 20-13）。选中此项后，虚拟机的网卡直接被桥接到网络中，

虚拟机将能完全地访问整个网络。

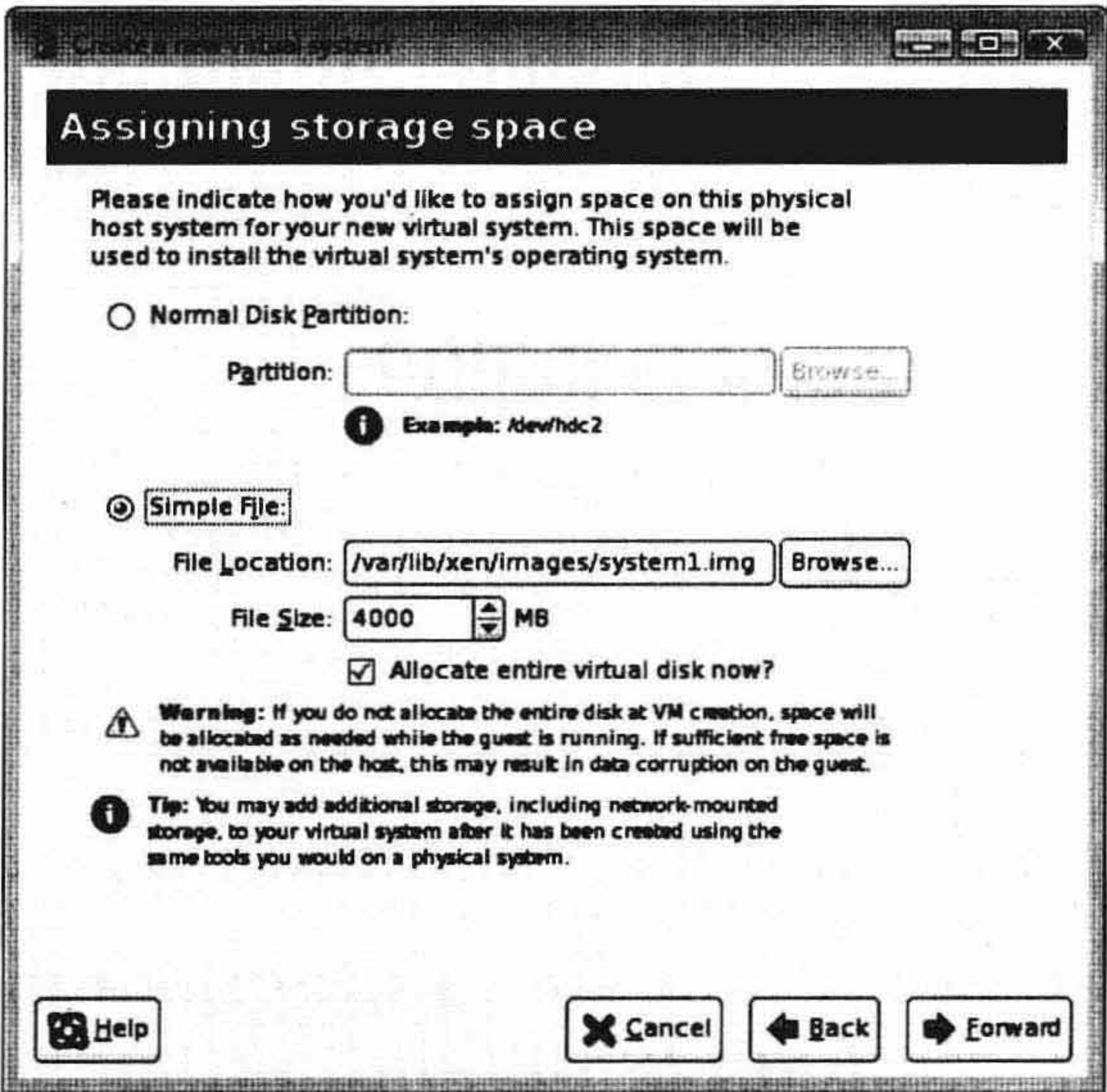


图 20-12 选择“Simple File”选项来创建存储后端

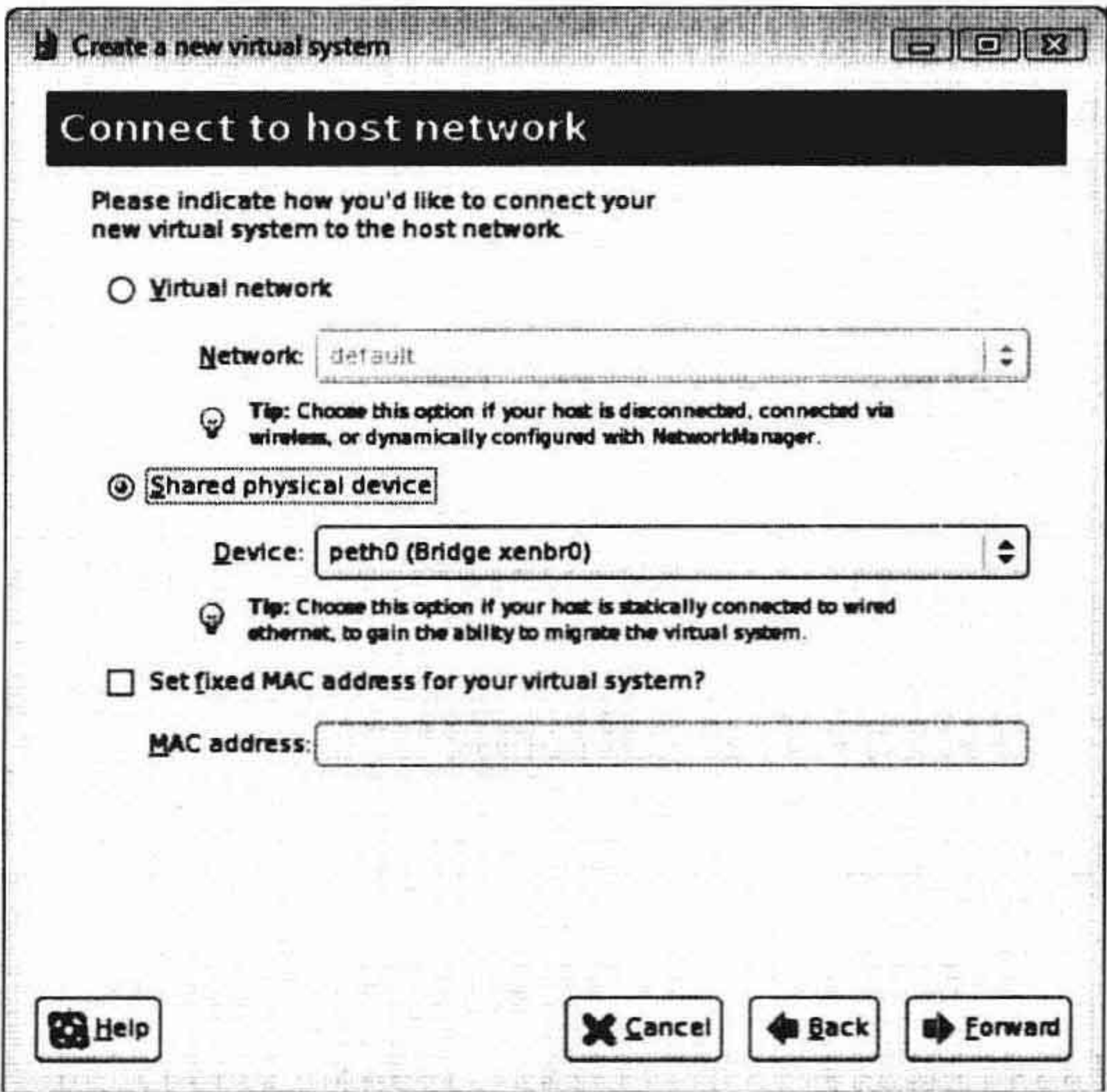


图 20-13 修改虚拟网络选项，使用“Shared physical device”

9. 接下来，需要设置分配给虚拟机的内存（见图 20-14）。由于可以动态调整分配给虚拟机的内存，因此，必须指定一个最大内存分配数量和一个最小内存分配数量。建议将最大内存分配数量设置为远高于默认内存分配数量，以便于在后面的使用过程中不用关闭虚拟机就可以动态增加其可用内存。

10. 这时，可以看到以上所有设置的概览页面。在此页面上单击“Forward”开始安装虚拟机。

到此为止，就安装了一个 Xen 虚拟机。如果读者按照前述步骤中给出了各种参数，就应该已经创建了一个用作存储后端的磁盘文件和一个配置文件。后面还应该多关注一下这个配置文件。

当在 Xen 中创建虚拟机时，会同时为此虚拟机创建两个配置文件。此例中，创建的虚拟机名为 sles10。如果仅仅是为了测试的目的，要在 Novell 机上看一下 SUSE 企业版 Linux，那么创建一个虚拟机是最好的方法了。由于使用的 SUSE 企业版 Linux 服务器版本为 10，虚拟机的名字就取作 sles10。同时会创建两个虚拟机配置文件/etc/xen/vm/sles10 和/etc/xen/vm/sles10.xml。在写作本书时，后一个配置文件还没有启用，启用的是前面的配置文件，其中包含了虚拟机的全部配置。列表 20-3 显示了此配置文件的内容。

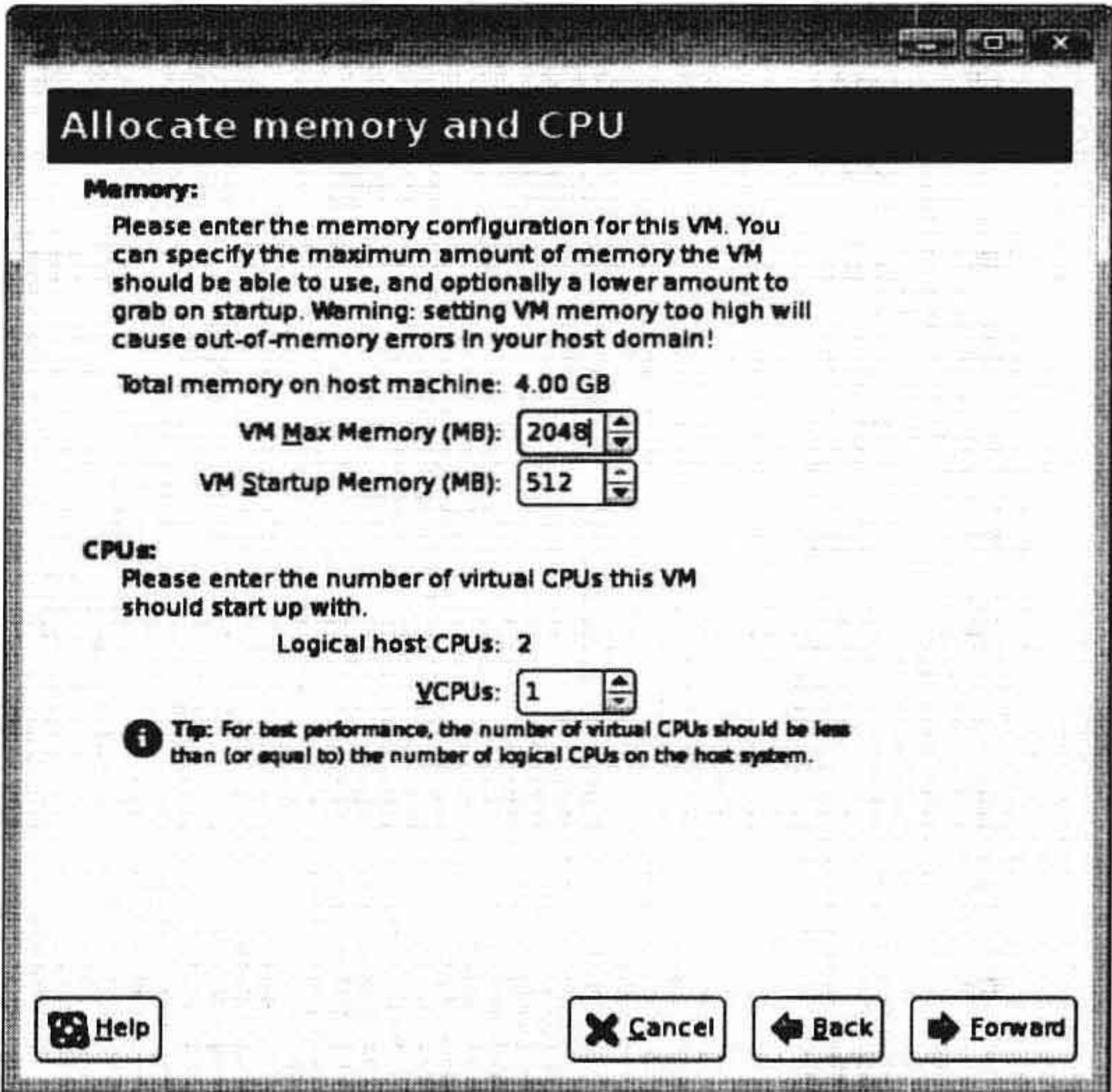


图 20-14 为虚拟机分配内存和 CPU

列表 20-3 虚拟机配置文件的内容

```
HKG:~ # cat /etc/xen/vm/sles10
name="sles10"
uuid="cbadcde6-f744-3dae-e3d8-f4c6a364fe85"
memory=512
vcpus=4
on_poweroff="destroy"
on_reboot="restart"
on_crash="destroy"
localtime=0
builder="linux"
bootloader="/usr/lib/xen/boot/domUloader.py"
bootargs="--entry=xvda2:/boot/vmlinuz-xen,/boot/initrd-xen"
extra=" "
disk=[ 'file:/var/lib/xen/images/sles10/disk0,xvda,w',
       'file:/isos/sles102.iso,xvdb:cdrom,r', ]
vif=[ 'mac=00:16:3e:6b:d0:a4', ]
vfb=[ 'type=vnc,vncunused=1']
```

如果仔细阅读了配置文件中的内容，可以看到其中的一些设置就是安装虚拟机时使用虚拟机管理器配置的内容。下面对文件中的参数作一个简单的解释。

- **name:** 虚拟机的名称。
- **uuid:** 分配给虚拟机的通用唯一标识符 (UUID)。
- **memory:** 目前分配给虚拟机的 RAM 总数。
- **vcpus:** 分配给虚拟机的 CPU 数量。
- **on_poweroff:** 定义当关闭运行着虚拟机的物理机时如何对虚拟机进行处理。设置为 **destroy** 意思是关闭虚拟机。
- **on_reboot:** 定义当重启运行着虚拟机的物理机时如何对虚拟机进行处理。
- **on_crash:** 定义当运行着虚拟机的物理机崩溃时如何对虚拟机进行处理。
- **localtime:** 若虚拟机使用 UTC 则将其设为 0，若使用本地时间则设为 1。
- **builder:** 指出使用什么操作系统创建的此虚拟机。
- **bootloader:** 这是虚拟机中 GRUB 的代替品。虚拟机中没有自己的 boot loader，只能使用在此处指定的程序来代替 boot loader。
- **bootargs:** 给前面指定的 boot loader 程序传递附加的参数。
- **extra:** 为 boot loader 指派附加的选项。
- **disk:** 为虚拟机指定使用什么作为存储后端。此例中，两个磁盘设备都使用 **file** 作为存储后端。也有时会使用 **phy**，意思是一个物理设备。
- **vif:** 为虚拟网卡指定 MAC 地址和其他属性。
- **vfb:** 指定以何种方式访问虚拟机（若允许访问）。此例使用默认设置（即 VNC 方式）访问虚拟机。

20.3.3 管理 Xen 虚拟机

虚拟机有两种管理界面。部分基本的管理任务可以通过虚拟机管理器进行，要进行高级

管理，就需要用到 `xm` 命令了。后面将对两种方法分别进行介绍。

1. 使用虚拟机管理器管理虚拟机

虚拟机管理器界面中提供了有限的一些选项来对 Xen 虚拟机进行管理。可以通过在虚拟机管理器主界面上选中虚拟机来访问这些选项，这些选项大多和硬件管理有关。当选中一个虚拟机时，默认情况下可以看到显示当前 CPU 和内存利用率信息的页面（如图 20-15 所示）。

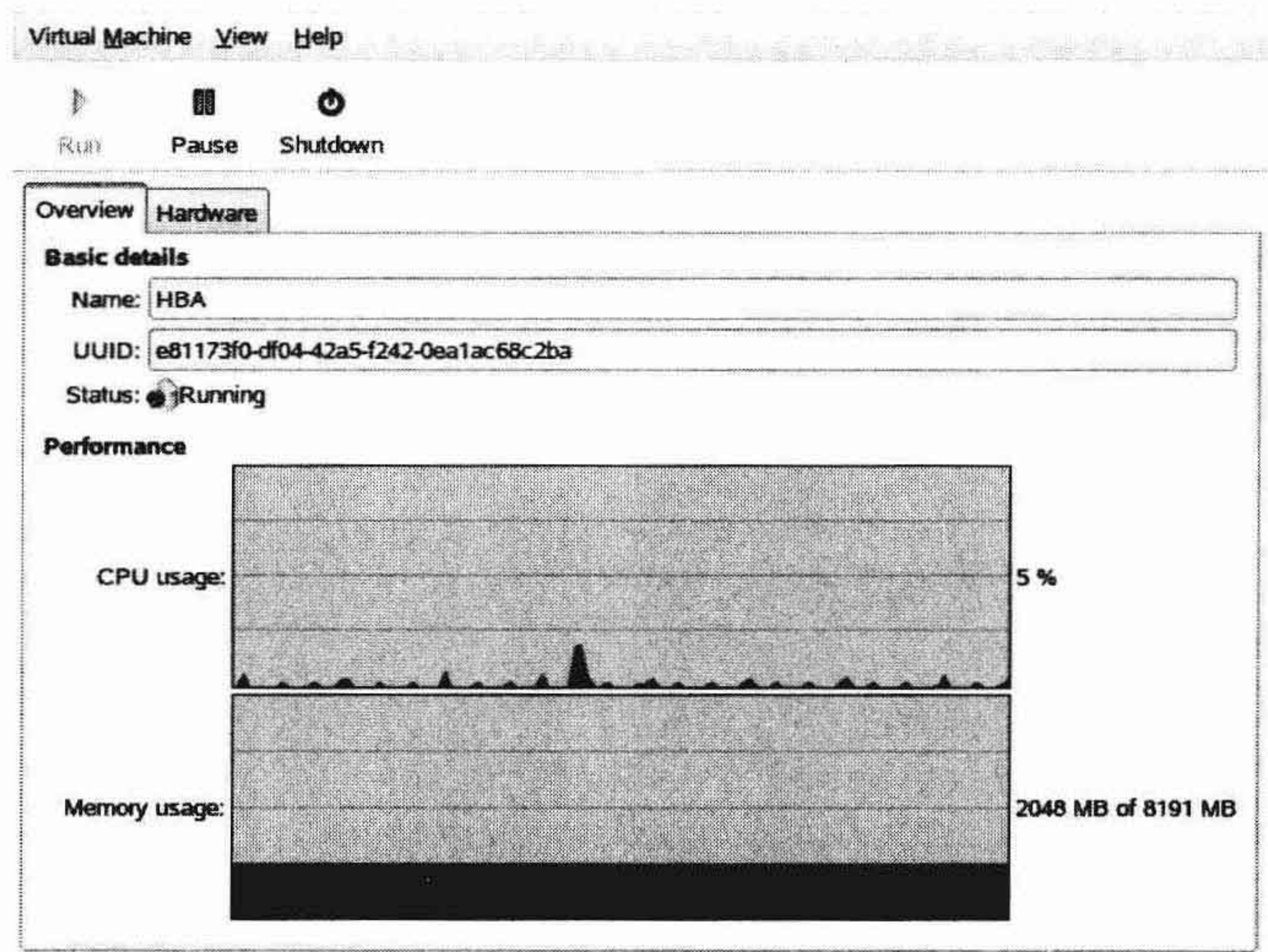


图 20-15 虚拟机概况窗口

若选中“Hardware”选项卡，可以看到目前分配给虚拟机的硬件设置情况。特别是如果使用的是一个半虚拟化的虚拟机，还会有很多非常有用的选项可以使用。比如，可以不用重启虚拟机而更改其 CPU 的数量以及目前分配的内存数量。要更改目前分配的内存数量，在“Hardware”选项卡上选中“Memory”。界面上将会显示出已经分配的内存数量。输入新分配的数值并单击“Apply”就可以修改内存分配量（如图 20-16 所示）。然后，虚拟机将立即应用此设置。

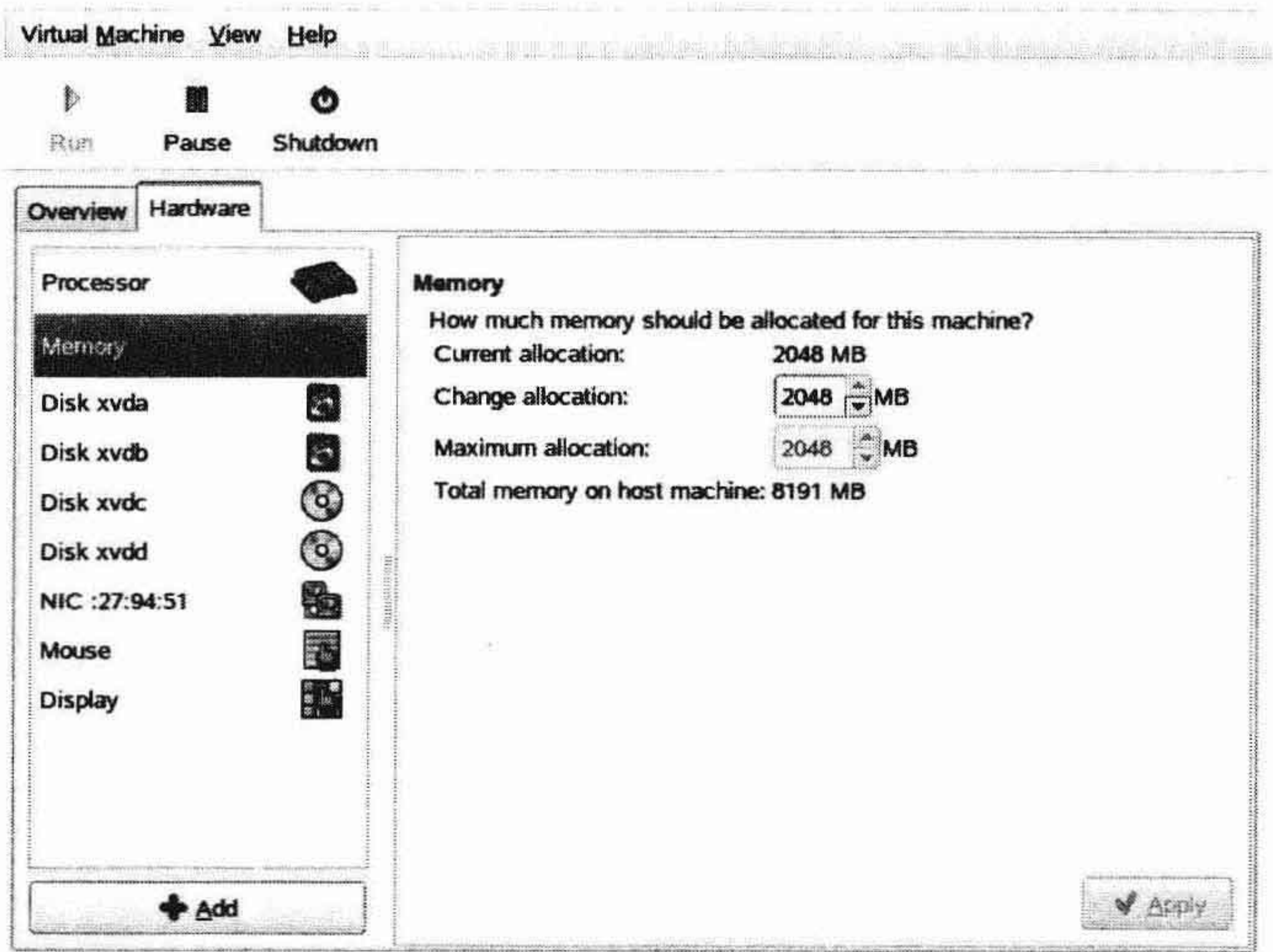


图 20-16 为半虚拟化的虚拟机更改内存分配

在虚拟机管理器硬件概况界面上还可以添加新的硬件。但是并不是所有的配置都可以立即写入配置文件，也就是说必须重启虚拟机以应用新设置。在当前界面上可以分配新的存储设备、网卡以及显卡等设备。单击“Add”按钮会开启一个向导，此向导将会引导用户完成添加新硬件的整个过程。

2. 使用 xm 命令管理虚拟机

虚拟机管理器提供了很多基本的管理选项。若想释放出虚拟机完全的管理能力，就需要用到 xm 命令。此命令带有多项不同参数，可以非常准确地告诉管理程序对虚拟机执行何种操作。如列表 20-4 所示，xm help 命令给出了所有可用选项的完整概况，其中每一个选项都可以在命令行中使用“xm <option> help”命令获取帮助，获得更多的有用信息。

列表 20-4 xm 所有可用选项的完整列表

```
HKG:~ # xm help
Usage: xm <subcommand> [args]

Control, list, and manipulate Xen guest instances.

xm full list of subcommands:

console      Attach to <Domain>'s console.
create       Create a domain based on <ConfigFile>.
new          Adds a domain to Xend domain management
delete       Remove a domain from Xend domain management.
destroy      Terminate a domain immediately.
domid        Convert a domain name to domain id.
domname      Convert a domain id to domain name.
dump-core    Dump core for a specific domain.
list         List information about all/some domains.
mem-max      Set the maximum amount reservation for a domain.
mem-set      Set the current memory usage for a domain.
migrate      Migrate a domain to another machine.
pause        Pause execution of a domain.
reboot       Reboot a domain.
rename       Rename a domain.
restore      Restore a domain from a saved state.
resume       Resume a Xend managed domain
save         Save a domain state to restore later.
shutdown    Shutdown a domain.
start        Start a Xend managed domain
suspend      Suspend a Xend managed domain
sysrq       Send a sysrq to a domain.
trigger      Send a trigger to a domain.
top          Monitor a host and the domains in real time.
unpause      Unpause a paused domain.
uptime       Print uptime for all/some domains.
vcpu-list    List the VCPUs for all/some domains.
vcpu-pin     Set which CPUs a VCPU can use.
vcpu-set     Set the number of active VCPUs for allowed for
              the domain.
debug-keys   Send debug keys to Xen.
dmesg        Read and/or clear Xend's message buffer.
info         Get information about Xen host.
log          Print Xend log
```


serve	Proxy Xend XMLRPC over stdio.
sched-credit	Get/set credit scheduler parameters.
sched-sedf	Get/set EDF parameters.
block-attach	Create a new virtual block device.
block-detach	Destroy a domain's virtual block device.
block-list	List virtual block devices for a domain.
block-configure	Change block device configuration
network-attach	Create a new virtual network device.
network-detach	Destroy a domain's virtual network device.
network-list	List virtual network interfaces for a domain.
vtpm-list	List virtual TPM devices.
vnet-list	List Vnets.
vnet-create	Create a vnet from ConfigFile.
vnet-delete	Delete a Vnet.
labels	List <type> labels for (active) policy.
addlabel	Add security label to domain.
rmlabel	Remove a security label from domain.
getlabel	Show security label for domain or resource.
dry-run	Test if a domain can access its resources.
resources	Show info for each labeled resource.
dumppolicy	Print hypervisor ACM state information.
setpolicy	Set the policy of the system.
resetpolicy	Set the policy of the system to the default policy.
getpolicy	Get the policy of the system.
shell	Launch an interactive shell.

<Domain> can either be the Domain Name or Id.
For more help on 'xm' see the xm(1) man page.
For more help on 'xm create' see the xmdomain.cfg(5) man page.

使用 `xm` 命令时，最主要的命令是“`xm list`”，此命令可以列出目前在主机上存在的所有虚拟机。由于 `xm` 的其他命令都需要用到虚拟机的 ID 或名称，因此对 `xm` 命令的使用都应该从“`xm list`”命令开始。列表 20-5 显示了此命令运行结果的一个示例。

列表 20-5 使用 `xm list` 命令列出所有的虚拟机

```
HKG:~ # xm list
```

Name	ID	Mem	VCPUs	State	Time(s)
Domain-0	0	5497	4	r-----	9253.3
HBA	1	2048	4	-b----	69064.1
sles10	4	512	4	-b----	107.8

如列表 20-5 所示，每一个虚拟机的名称、ID、内存分配、分配的虚拟 CPU 数量、运行时间以及目前状态都列了出来。大多数情况下虚拟机处于两种状态：活动状态，意思是使用“`xm list`”命令之前虚拟机已经运行了一段时间；或者是阻塞状态，意思是使用此命令时虚拟机正等待 I/O 操作。

若“`xm list`”命令没有给出足够的细节信息，还可以使用“`xm top`”命令。列表 20-6 列出了此命令的运行结果。

列表 20-6 使用 `xm top` 命令查看虚拟机运行情况

```
HKG:~ # xm top
xentop - 20:20:20  Xen 3.2.0_16718_14-0.4
```



```
3 domains: 1 running, 2 blocked, 0 paused, 0 crashed, 0 dying, 0 shutdown
Mem: 8387960k total, 8383208k used, 4752k free CPUs: 4 @ 2666MHz
NAME STATE CPU(sec) CPU(%) MEM(k) MEM(%) MAXMEM(k) MAXMEM(%) VCPUS
NETS NETTX(k) NETRX(k) VBDS VBD_OO VBD_RD VBD_WR SSID
Domain-0 -----r 9267 0.0 5628928 67.1 no limit n/a 4
0 0 0 0 0 0 0 2148573580
HBA --b--- 69078 0.0 2097152 25.0 2097152 25.0 4
1 87381755 207537763 4 3 20436185 52349866 2148573580
sles10 --b--- 108 0.0 524288 6.3 524288 6.3 4
1 3 1653 2 0 34211 20867 2148573580
```

```
Delay Networks vBds VCPUs Repeat header Sort order Quit
```

就像 Linux 系统工具 top 一样，“xm top”命令显示出了虚拟机的详细运行信息。命令每 5 秒钟刷新一次其自身的状态，使用户可以随时观察虚拟机运行过程的变化。读者应该已经注意到了，“xm top”命令的输出信息中每个虚拟机占两行。其中特别有用的是网络相关的信息，NETS 参数显示了虚拟机使用的虚拟网络接口数量，NETTX 和 NETRX 参数给出了网卡发送和接收到的数据块数量。在列表 20-6 所示的示例中，可以看到虚拟机 HBA 是目前在网络中最活跃的机器。

若物理机发生了故障，可能需要按下电源按钮来快速关机。在虚拟机中则可以使用 xm 命令的“xm destroy”来做同样的事情，此命令一般带有虚拟机的名称或 ID 作为参数。比如，“xm destroy 4”命令将立即关闭 xm 列表中 ID 为 4 的虚拟机。别担心，这并不会破坏什么，还可以很容易地使用命令“xm start sles10 启”动虚拟机。

20.3.4 自动启动 Xen 虚拟机

当重启物理机时，用户可能会希望其中的虚拟机也能随主机一起启动。做到这点并不难，以下步骤描述了设置的过程。

- 1. 为确保 xendomains 服务自动启动，可以将其加入到 runlevel 中。如，命令“chkconfig --add xendomains”就可以做到这一点。
- 2. 使用命令“mkdir /etc/xen/vm/auto”创建名为/etc/xen/vm/auto 的目录。
- 3. 从/etc/xen/vm 目录中找出用来启动虚拟机的配置文件。比如，像前面讨论过的，要使在文件 sles10 中配置的虚拟机自动启动，就需要 sles10 文件，而且需将此文件复制到刚刚创建的目录/etc/xen/auto 下。

Xen 还提供了很多其他特性，可以参考发行版中提供的文档以获取更多信息。

20.4 使用 KVM 安装虚拟机

如果主机中的 CPU 支持虚拟化技术，基于 KVM 的虚拟化是最容易实现的，要做的事情只是加载一个内核模块而已。本章将会讨论如何为实现 KVM 虚拟化准备主机环境，然后讨论如何在 KVM 虚拟化环境中安装 Windows 和 Ubuntu 虚拟操作系统。因为 Ubuntu 服务器是

第一个应用 KVM 的企业 Linux 发行版，本章就使用 Ubuntu 服务器作为示例操作系统。

■警告：使用虚拟化环境时，最好将主机操作系统和其他系统加以区分。主机操作系统是服务器启动的第一个操作系统，而且此系统还要为其他操作系统负一些特别的责任，比如管理驱动器访问以及对虚拟机自身进行管理。为了确保它能够以最有效的方式完成这些任务，尽量不要在主机操作系统中运行任何服务（除了虚拟化服务）。

20.4.1 为 KVM 虚拟化准备服务器：网络方面

在使用虚拟机的服务器中，安装的虚拟机的数量可以比物理网卡的数量多得多。因此，需要将虚拟化方案配置为各个虚拟机共享服务器的网卡。要实现这个目的，需要创建一个虚拟网络桥，若想要一切顺利，最好在配置第一个虚拟机之前就这样做。

开始之前，确保 bridge-utils 软件包已经安装。接下来，创建网桥，配置/etc/network/interface 文件中的配置，如列表 20-7 所示，使用此段代码代替文件中现有的所有内容。

列表 20-7 创建网桥

```
auto lo
iface lo inet loopback
auto br0
iface br0 inet static
    address 192.168.1.99
    network 192.168.1.0
    netmask 255.255.255.0
    broadcast 192.168.1.255
    gateway 192.168.1.254
    bridge_ports eth0
    bridge_fd 0
    bridge_hello 2
    bridge_maxage 12
    bridge_stop off
```

此配置文件确保重启服务器或网络服务时，将创建一个名为 br0 的设备来代替 eth0 设备。此设备作为一个虚拟网桥运行，所有的虚拟机网卡都绑定到此设备。不过，此设备的物理后端是物理网卡 eth0，此项由“bridge ports eth0”行指定。如此创建配置文件之后，使用“sudo invoke-c.d networking restart”命令重启网络服务。现在就准备好处理 KVM 虚拟机了。

20.4.2 在 Ubuntu 服务器上设置 KVM

根据如下步骤为服务器设置虚拟化（此处叙述的步骤支持 Ubuntu 服务器 8.04 版本及更新版本）。

1. 为 KVM 虚拟化安装所有必须的软件（主要是 KVM 和 QEMU 软件包）。root 用户可以使用命令“apt-get install kvm qemu libvirt-bin”安装。
2. 安装完所有软件包后，确保已经加载 kvm 内核模块。可以使用 lsmod 命令进行检测（ls |grep kvm），如果模块没有加载，可以使用“modprobe kvm”安装。

■提示：加载内核模块时是不是收到了“操作不支持”的提示？如果是，则表示主机的 CPU 不支持。要么升级一个提供虚拟化支持的 CPU，或者使用 Xen 虚拟化解决方案来创建一个半虚拟化的操作系统。

接下来，还要做一些准备工作，其中就包括设置用来创建虚拟机的 libvirt 工具。首先，需要在 libvirtd 组中添加一个可以管理 KVM 的用户账号。可以使用命令“`sudo adduser <username>`”添加一个用户，使用要添加的用户账号名替换<username>就可以了。

好了！Ubuntu 服务器现在就已经准备好安装及配置虚拟操作系统了。下一节将讨论如何安装一个 Windows 虚拟操作系统。

20.4.3 在 KVM 上安装一个 Windows 客户端操作系统

在把 Windows 作为第一个客户端操作系统安装时，应该自己先弄清楚自己到底要使用这个虚拟机来做什么。服务器是在一个数据中心的运行吗？需要远程完成所有的任务（包括虚拟机的安装）吗？如果是如此，可以不使用图形界面的 GUI，而是在工作站中使用命令的方式完成。如果不是如此，而是要在物理机服务器中完成虚拟机的管理工作，则最好在服务器中安装一个 GUI 工具。本节中描述的步骤假定用户已经安装了某个可以用来显示 Windows 安装界面的图形化界面工具。

要想安装一个 Windows 虚拟操作系统，首先应该设置存储设施。测试虚拟化的最简单方法就是使用一个磁盘映像文件，可以使用 `dd` 或 `qemu-img` 命令创建一个磁盘映像文件，如下命令所示，在目录 `/var/lib/virt`（应确保在创建映像文件之前创建此目录）下创建了一个大小为 8GB、名为 `windows.img` 的磁盘映像文件。

```
sudo dd if=/dev/zero of=/var/lib/virt/windows.img bs=1M count=8192
```

现在已经创建了一个磁盘镜像文件，可以使用 `kvm` 命令来安装 Windows 了。把 Windows 安装 CN 放在驱动器中（或者使用一个 ISO 文件），然后使用如下命令来开始安装，创建一个使用 512MB 内存的 Windows 虚拟机，此命令中用到了刚刚创建的 `windows.img` 磁盘文件。如果要使用一个 ISO 文件来代替物理光驱，只要使用 ISO 文件的完整路径代替命令中的 `/dev/cdrom` 就可以了。

```
kvm -m 512 -cdrom /dev/cdrom -boot d windows.img
```

■提示：`kvm` 是否警告 CPU 缺少虚拟化支持呢？可能是没有在系统的 BIOS 中打开对虚拟化的支持。重启主机，进入系统 BIOS，开启虚拟化支持。一般地，相关的设置项应该在 BIOS 配置中的 Advanced 一节，虚拟化支持项应该有一个类似 `vm`、`vt` 或 `virtualization` 的名称。

上述操作将开启一个 QEMU 窗口，在此窗口中可以看到 Windows 安装器的加载，可以从此界面中完成 Windows 安装。

一旦虚拟化 Windows 安装完成，就可以使用安装时同样的方法来运行它了。还是使用 `vm` 命令，但是要省略“-boot d”选项，这样以确保先从 CD-ROM 中启动。如下命令运行了

安装在 windows.img 文件中的 Windows 实例。

```
kvm -m 512 -cdrom /dev/cdrom windows.img
```

现在有了一个 Windows 虚拟机。安装的过程很简单吧？接下来，还要看一下如何在 Ubuntu 服务器虚拟化主机上安装一个 Ubuntu 客户机。

20.4.4 在 KVM 上安装一个 Ubuntu 服务器客户端操作系统

阅读了前面章节中关于如何在 KVM 上安装一个 Windows 客户机操作系统的内容后，读者应该能够猜出如何安装一个 Ubuntu 虚拟机实例了吧。基本上，安装 Windows 和安装 Ubuntu 没有什么区别，即创建一个虚拟磁盘，然后在其上安装一个 Ubuntu 服务器。准备好一个 Ubuntu 安装媒体的 ISO 文件，运行如下命令。

1. 创建磁盘文件。

```
dd if=/dev/zero of=/var/lib/virt/ubuntu.img bs=1M count=4096
```

2. 使用 kvm 命令从 Ubuntu ISO 文件中开始安装。

```
kvm -m 256 -cdrom /isos/ubuntu.iso -boot d /var/lib/virt/ubuntu.img
```

提示：在安装 Ubuntu 或其他 Linux 发行版时有什么问题吗？原因可能是在于如今在安装之前出现的启动加载图形化界面。尝试使用一个类似于 Ubuntu 网络启动时 mini.iso 文件的非图形化安装程序来代替。此种方式可以无任何问题的安装所有的 Linux 发行版。

3. 就像安装一个一般的服务器一样安装 Ubuntu。
4. 使用如下命令启动刚刚安装的虚拟化 Ubuntu 服务器。

```
kvm -m 256 ubuntu.img
```

20.4.5 使用虚拟机管理器管理 KVM 虚拟机

如果不喜欢使用 kvm 命令（或甚至是此命令的加强版，如“virt-install:sudo apt-get install python-virtinstall”）安装和启动虚拟机，则可以使用 virtual-manager。此图形化工具提供了安装和管理虚拟机的一个简单方法。但它也有一个缺点，那就是需要运行在 X 服务器中。这并非意味着一定要在服务器中安装一个图形化环境，也可以在工作站中运行 virt-manager 命令来代替，比如，从工作站中，与服务器建立一个 SSH 会话并启动一个 virt-manager。这样做既能使用图形化虚拟机管理工具带来的方便，又省去了在服务器上安装 GUI 的麻烦。作为替代方案，也可以使用启动参数告诉 virt-manager 必须从另一台计算机中获取信息。例如，下述命令连接到了 somenode.example.com 并允许从此节点处管理虚拟机。

```
virt-manager -c qemu+ssh://somenode.example.com/system
```

如果要在本机上启动 virt-manager 来创建虚拟机，可以使用如下命令。

```
virt-manager -c qemu:///system
```


在如下的步骤中，读者可以学到如何使用 virt-manager 命令创建一个虚拟机。

1. 启动 virt-manager。此例中，假定读者已经与远程服务器建立了 SSH 连接，那么便可以输入“virt-manager -c qemu:///system”命令启动 virt-manager 会话。另外，在 KVM 界面上，可以使用“File”➤“Connet”选项从本机上连接到 QEMU，如此便可以显示一个界面，如图 20-17 所示。

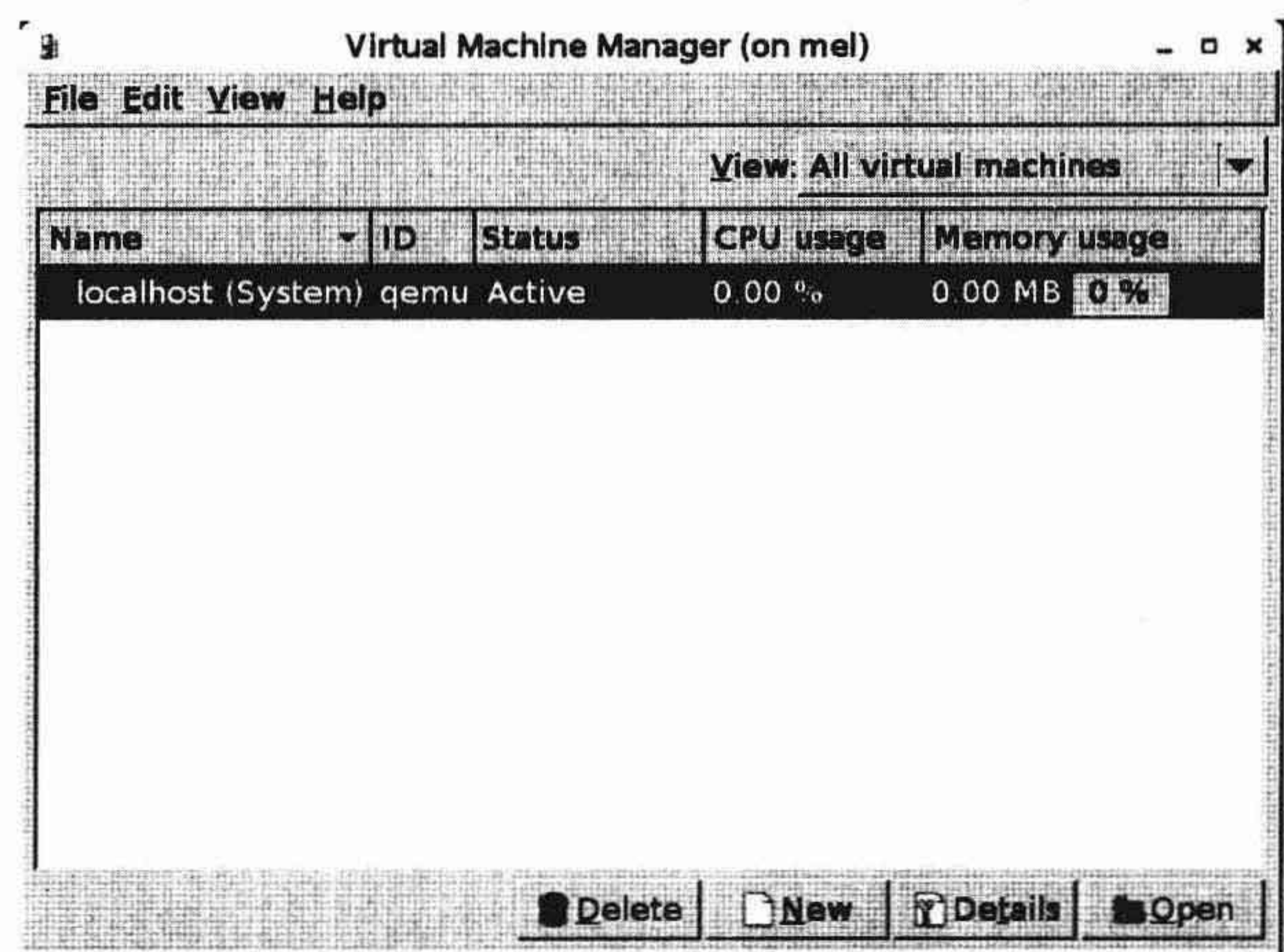


图 20-17 虚拟机管理

2. 选中“localhost”行并单击“New”开始创建一个新的虚拟机。此时会打开向导第一步界面，在此界面中单击“Forward”。
3. 输入要创建的系统的名称。比如，如果要安装一个 Windows XP 测试机，则可以使用 WinXP 作为系统的名称。接下来，单击“Forward”继续。
4. 工具询问虚拟机的类型。在 KVM 中，可以只使用此处显示的“Fully Virtualized”选项，“Paravirtualized”选项只在 Xen 虚拟化方案中使用（见图 20-18）。

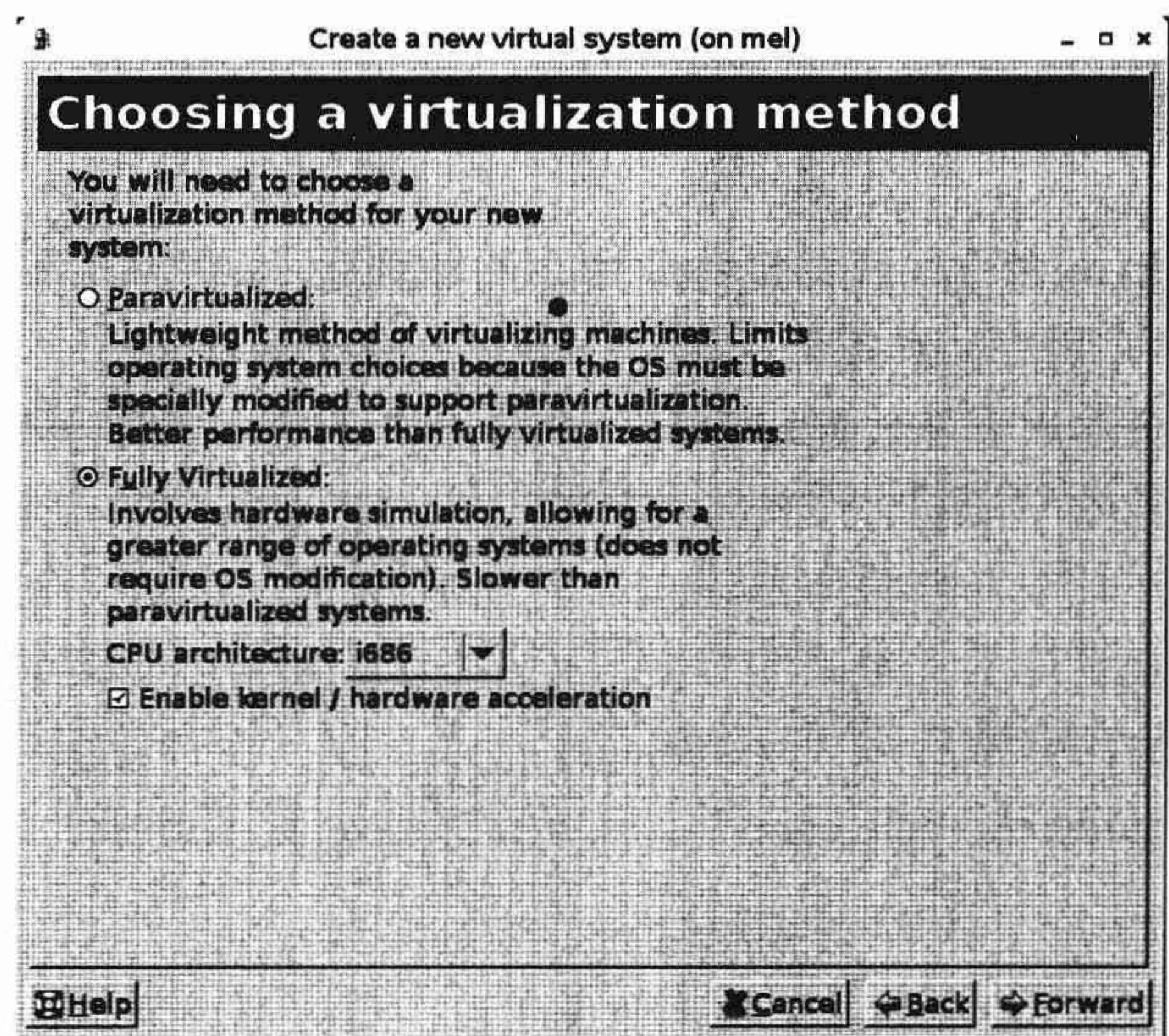


图 20-18 在 KVM 中，“Fully Virtualized”是唯一可用的选项

5. 现在需要指定如何启动安装。比如，如果有一个 ISO 镜像可以使用，则定位到可以找到 ISO 镜像的路径（见图 20-19），还要确保选中了正确的 OS 类型和 OS 变量。

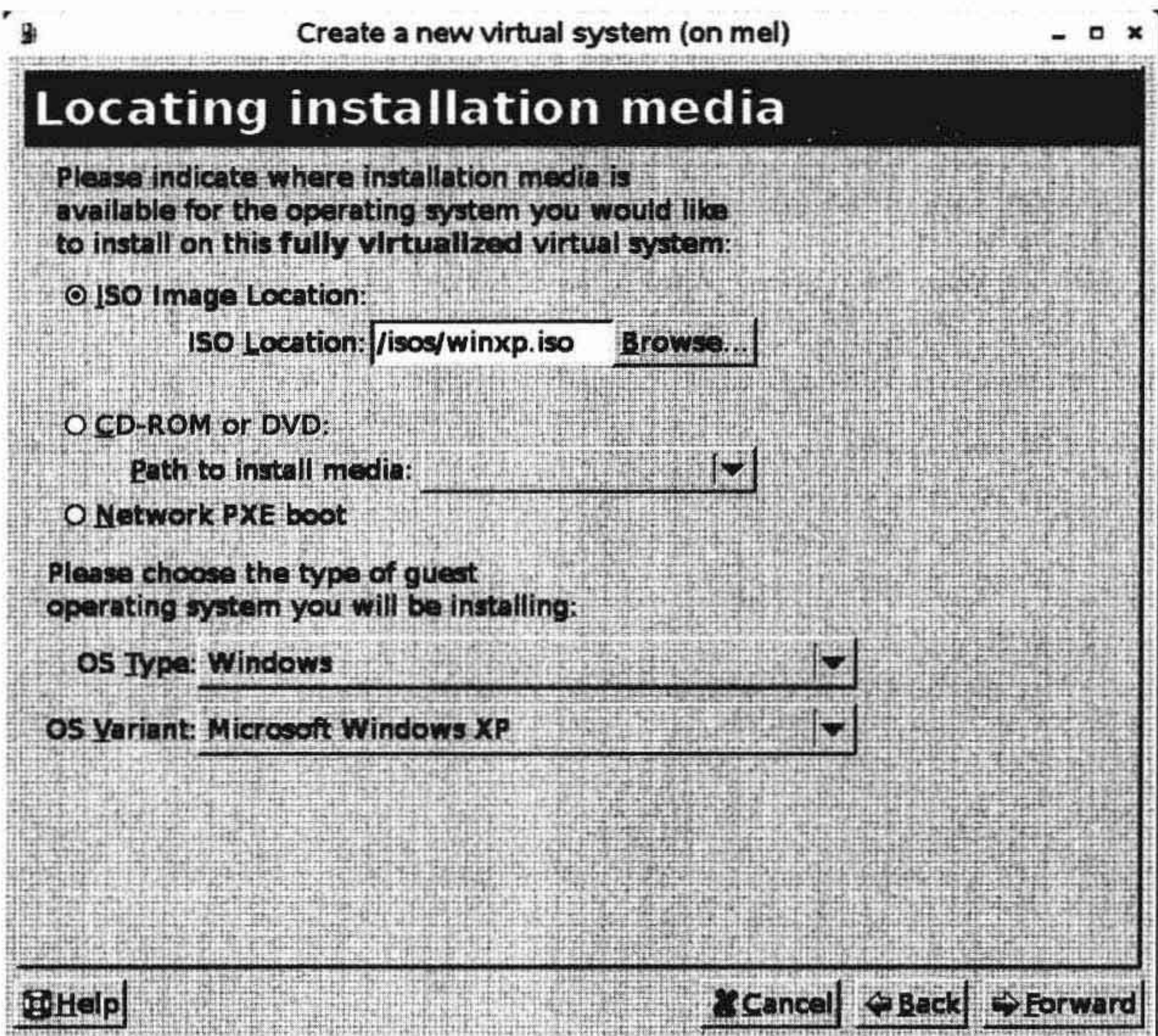


图 20-19 从 ISO 镜像文件中安装

6. 下一个窗口中，可以指定要安装何种系统（见图 20-20）。为了获得最好的性能，最好给每个虚拟机一个专用分区或 LVM 逻辑卷。如果不能这样做，可以将系统安装到一个文件中。安装界面可以自动创建此虚拟硬盘文件。

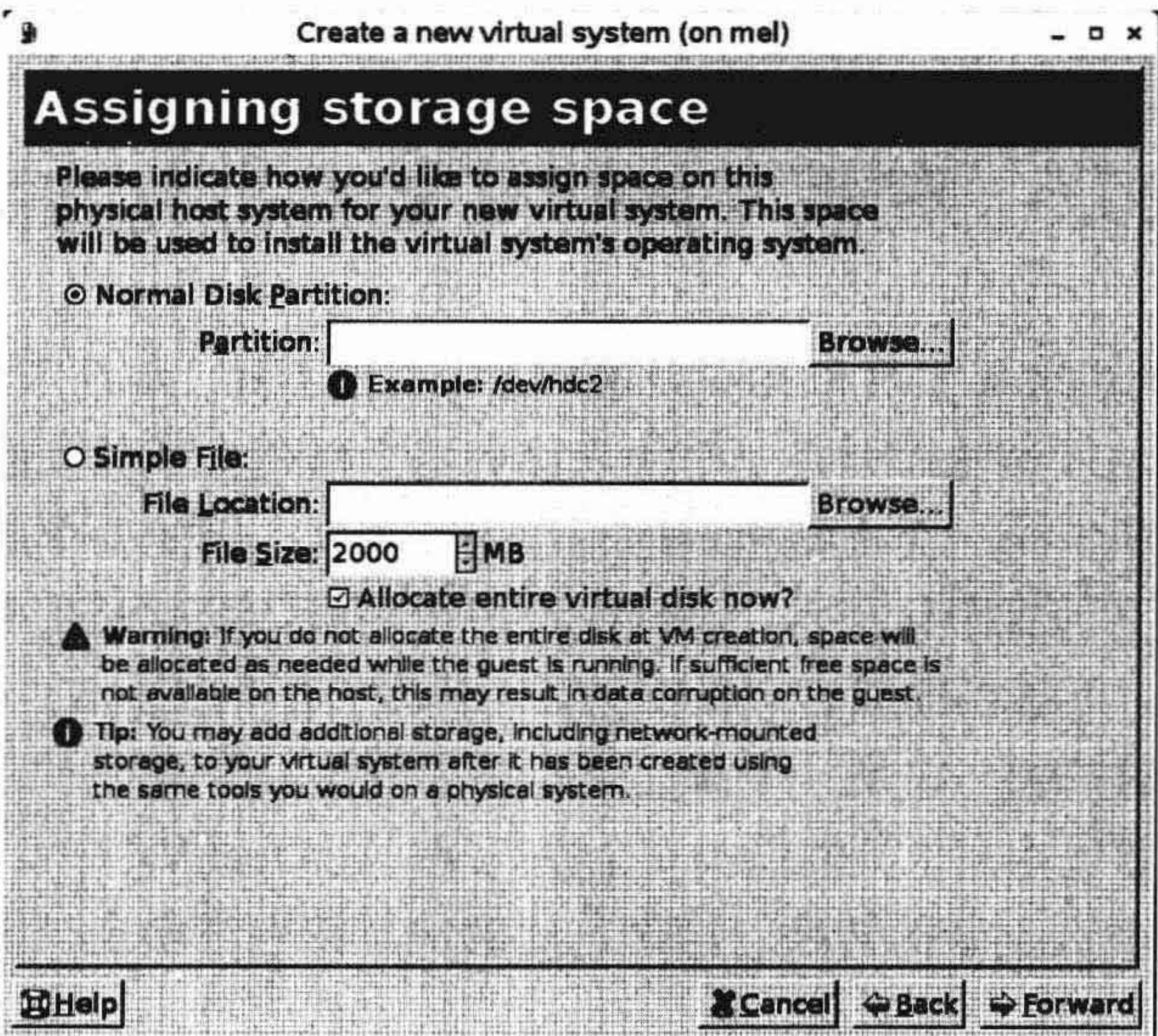


图 20-20 使用分区或 LVM 逻辑卷作为存储终端

7. 选择要使用的网络连接方式，最灵活的方式是创建一个虚拟网络。在此配置中，使用前面提到过的网桥方式总是适用的。如果服务器使用的是固定 IP 地址，还可以选择通过选中

“Shared physical device” 选项给网络接口分配第二个 IP 地址。完成后，单击 “Forward” 继续安装。

8. 现在输入要给虚拟机分配的内存数量和 CPU 数目，并单击 “Forward”（见图 20-21）。

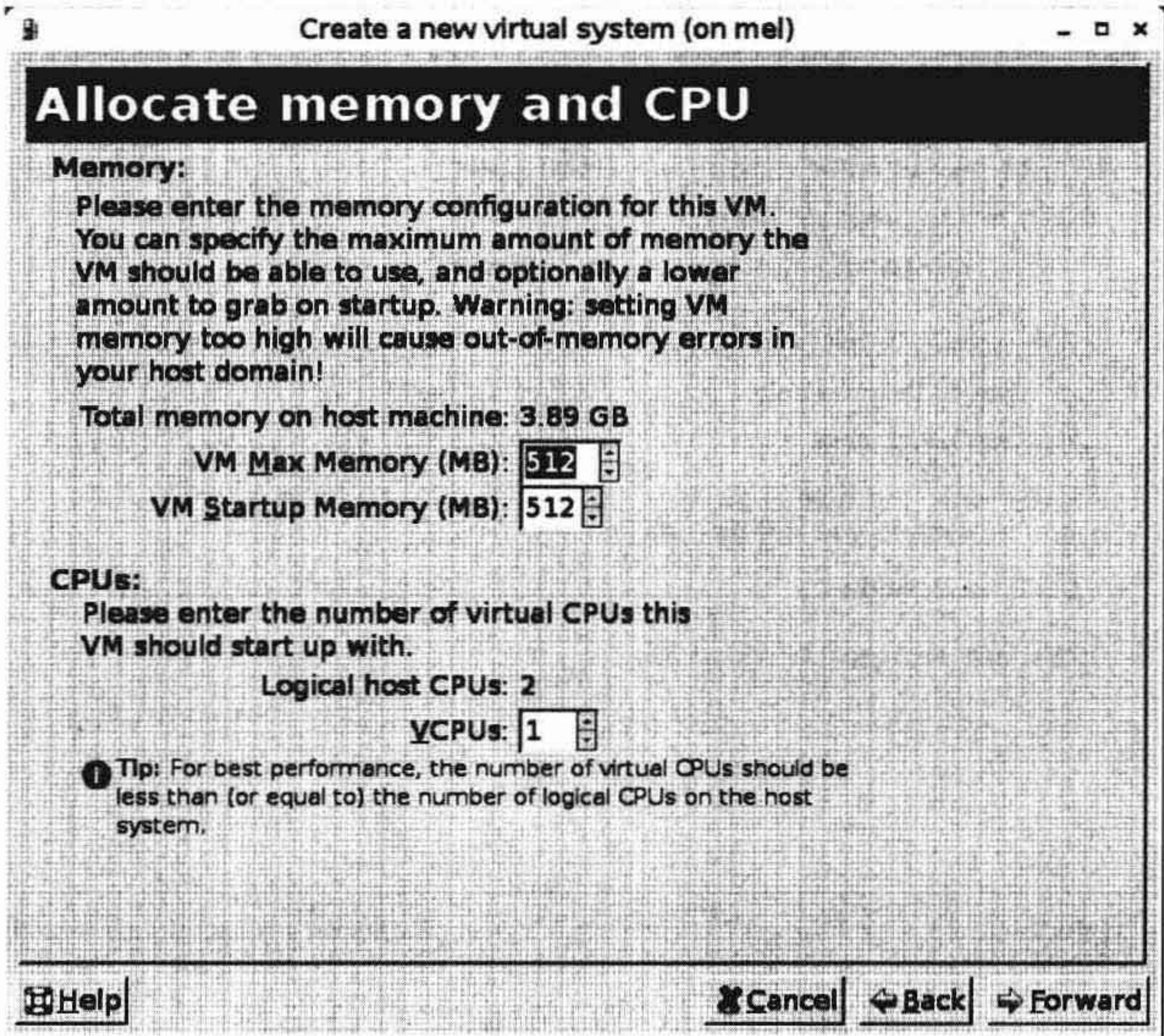


图 20-21 设置虚拟机中的内存和 CPU

9. 安装向导的最后一个页面中，可以看到已经做好的安装设置概览。对此设置还满意吗？若满意则单击 “Forward” 开始虚拟操作系统的安装。一旦安装完成，就可以看到新的虚拟机已经被添加到了 virt-manager 的界面上，并可以对其进行管理了。

20.5 使用 OpenVZ 的虚拟化方案

OpenVZ 中使用的虚拟化方法和 Xen 及 KVM 中截然不同。如前所述，OpenVZ 提供了一种基于容器的方法，这就意味着要在一个物理服务器上创建多个独立的容器。管理员可以在这些容器中创建虚拟机。每一个虚拟机的运行都完全独立，因此各个虚拟机都能与其他虚拟机安全共处。相对于其他虚拟化技术，OpenVZ 有个局限，那就是主机以及其他虚拟操作系统都必须运行 Linux。可以在虚拟客户机中运行不同发行版的 Linux。OpenVZ 还是 Patallels Virtuozzo Containers 的基础，它是一个商业化的解决方案，使用的方式完全相同。

此种方法的最主要优点在于其性能及安全性表现。在 OpenVZ 中，相互交流的只有 Linux 系统，因此不需进行什么复杂的转换。同时，此解决方案也很安全，因为各个容器之间的运行环境相互独立。

20.5.1 安装

可以在 Red Hat 和 Debian 上运行 OpenVZ，目前，还不支持在 Ubuntu 中的安装。下面的

步骤展示了在基于 Red Hat 的系统上安装 OpenVZ 的过程。此过程支持适用于 Yum 的安装，也适用于 RPM 软件包方式的安装，本节中的安装过程介绍的是如何使用 Yum 安装 OpenVZ。由于性能的关系，建议读者在 64 位的硬件环境和 64 位的操作系统中运行 OpenVZ。此处的安装过程假设读者是在 64 位的 OS 中进行安装。

1. OpenVZ 有一个库文件。安装之前，先下载并将其放在 `/etc/yum.repos.d` 目录中。可以使用如下 `wget` 命令下载此库文件。

```
wget http://download.openvz.org/openvz.repo
```

此命令之后，还要导入属于此库文件的 GPG 码，如下所示。

```
rpm --import http://download.openvz.org/RPM-GPG-Key-OpenVZ
```

2. 接下来需要安装 OpenVZ 内核。有几种不同的内核，所选的内核依赖于所使用的硬件和需要创建的容器数量。首先，有一个 SMP 内核，此内核适用于对称多处理器环境中。它支持 4GB 内存和最多 10 到 20 个容器。其次是 `entnosplit` 内核，它使用物理地址扩展（PAE）并支持 64GB 内存和 10 到 30 个容器。最后是企业内核，它支持 SMP 和 PAE。如果要同时处理多于 20 到 30 个容器，这是最好的选择。当在 32 位操作系统中使用 OpenVZ 时，选择一个正确的内核很重要。如果使用的是 64 位的操作系统，使用 SMP 内核就足够了，因为在 64 位环境中不必再使用 PAE，来突破 4GB 内存使用限制。假设读者使用 64 位环境，则可以使用下述命令安装此内核。

```
yum install ovzkernel-smp
```

此时，OpenVZ 内核已经安装完毕并自动添加到了 GRUB 配置中。重启后，OpenVZ 内核将自动启动。

3. 重启之前，需要调整几个 `sysctl` 参数。可以通过确定在 `/etc/sysctl.conf` 文件中有列表 20-8 中所示配置做此配置。

列表 20-8 文件 `/etc/sysctl.conf` 中必需的优化参数

```
net.ipv4.ip_forward = 1
net.ipv6.conf.default.forwarding = 1
net.ipv6.conf.all.forwarding = 1
net.ipv4.conf.default.proxy_arp = 0
net.ipv4.conf.all.rp_filter = 1
kernel.sysrq = 1
net.ipv4.conf.default.send_redirects = 1
net.ipv4.conf.all.send_redirects = 0
```

4. 接下来，关闭 SELinux。可以在 `/etc/sysconfig/selinux` 文件中加入如下值来实现。

```
SELINUX=disabled
```

5. 安装 OpenVZ 工具。使用如下命令。

```
yum install vzctl vzquota
```

6. 如果要给虚拟机使用一个不同于主机 IP 地址的 IP 地址范围，需要修改文件 `/etc/vz/vz.conf` 并确保其中含有如下所示的行。


```
NEIGHBOUR_DEVS=all
```

7. 到此为止，可以重启主机以激活 OpenVZ 内核。

20.5.2 创建 OpenVZ 虚拟机

要创建一个 OpenVZ 虚拟机，先要准备想要创建的虚拟操作系统的模板。所有主要的 Linux 发行版都有可用的模板，可以从如下链接处下载一个模板的列表，见“<http://wiki.openvz.org/Download/template/precreated>”。比如，如果要使用一个 CentOS 5 虚拟机，需要使用下述命令下载 CentOS 5 模板。

```
cd /vz/template/cache
wget http://download.openvz.org/template/precreated/contrib/
centos-5-i386-default.tar.gz
```

下载到模板后，就可以使用它来启动一个或多个虚拟机了。可以使用如下命令启动一个基于刚刚下载的模板的虚拟机。

```
vzctl create 150 --ostemplate centos-5-i386-default --config vps.basic
```

其中，`vzctl` 是用来创建和管理虚拟机的主要命令。就像 `ip` 命令一样，`vzctl` 也用到了很多子命令，其中的 `create` 命令就用来创建一个新的虚拟机。每一个虚拟机都有自己的唯一的 ID。一个好方法是使用 IP 地址的最后部分来做这个唯一的 ID（示例中是 150）。接下来，需要指定使用哪一个模板，该模板上包含了一些需要填充的空配置项和基本文件。“`--config`”选项确保为虚拟机创建一个配置文件，可以在 `/etc/vz/conf` 目录下找到此配置文件。每一个虚拟机都有自己的配置文件，可以直接编辑此文件来管理虚拟机，还可以在命令行中传递不同的参数来更改虚拟机的属性。

用此法创建虚拟机后，它不会随主机自动启动。要使其自动重启，可以使用如下命令。

```
vzctl set 150 --onboot yes --save
```

接下来，可以开始加入在虚拟机中使用的其他参数。其中至少包括 IP 配置，配置命令如下所示。可以更改命令中的参数以匹配目前的配置。

```
vzctl set 150 --hostname nuuk.example.com --save
vzctl set 150 --ipadd 192.168.1.150 --save
vzctl set 150 --nameserver 193.79.237.39 --save
```

到目前为止，已经创建了一个相当不错的基本配置，并保存在文件 `/etc/vz/conf/150.conf`（假设示例中使用 150 作为虚拟机的 ID）中。列表 20-9 显示了到目前为止此文件的内容。

列表 20-9 写入到一个配置文件中的虚拟机配置

```
[root@centos conf]# cat 150.conf
# Copyright (C) 2000-2008, Parallels, Inc. All rights reserved.
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
```



```

#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
#
ONBOOT="yes"

# UBC parameters (in form of barrier:limit)
KMEMSIZE="14372700:14790164"
LOCKEDPAGES="256:256"
PRIVVMPAGES="65536:69632"
SHMPAGES="21504:21504"
NUMPROC="240:240"
PHYSPAGES="0:9223372036854775807"
VMGUARPAGES="33792:9223372036854775807"
OOMGUARPAGES="26112:9223372036854775807"
NUMTCPSOCK="360:360"
NUMFLOCK="188:206"
NUMPTY="16:16"
NUMSIGINFO="256:256"
TCPSNDBUF="1720320:2703360"
TCPRCVBUF="1720320:2703360"
OTHERSOCKBUF="1126080:2097152"
DGRAMRCVBUF="262144:262144"
NUMOTHERSOCK="360:360"
DCACHESIZE="3409920:3624960"
NUMFILE="9312:9312"
AVNUMPROC="180:180"
NUMIPTENT="128:128"

# Disk quota parameters (in form of softlimit:hardlimit)
DISKSPACE="1048576:1153024"
DISKINODES="200000:220000"
QUOTATIME="0"

# CPU fair sheduler parameter
CPUUNITS="1000"

VE_ROOT="/vz/root/$VEID"
VE_PRIVATE="/vz/private/$VEID"
OSTEMPLATE="centos-5-i386-default"
ORIGIN_SAMPLE="vps.basic"
HOSTNAME="nuuk.example.com"
IP_ADDRESS="192.168.1.150"
NAMESERVER="193.79.237.39"

```

现在，创建了一个虚拟机配置，也是时候使用如下命令启动虚拟机了。

```
vzctl start 150
```

接下来，设置 root 用户的密码，如下所示。

```
vzctl exec 150 passwd
```


`passwd` 将提示设置 `root` 用户的密码，就像从命令行中运行 `passwd` 命令时的情景一样。

到此为止，虚拟机容器已经准备好被使用了。可以使用 `SSH` 或直接从操作系统的控制台中输入如下命令连接到虚拟机。

```
vzctl enter 150
```

如果使用“`vzctl enter`”访问了虚拟机容器，可以通过输入“`exit`”来退出此命令。

20.5.3 OpenVZ 虚拟机基本管理

虚拟机已启动并正在运行，现在可以了解一下可以用来管理虚拟机的一些命令。可以看到，这些命令看起来就像用来管理 `Xen` 虚拟机的一样。首先来看一下目前可用的虚拟机管理命令列表，可以使用命令“`vzlist -a`”来显示此列表。

```
[root@centos ~]# vzlist -a
CTID      NPROC STATUS IP_ADDR      HOSTNAME
150       19 running 192.168.1.150 nuuk.example.com
```

可以看到，命令显示了每一个虚拟机目前的状态（此例中只有一个虚拟机在运行）。基于此信息，可以停止一个虚拟机（`vzctl stop 150`）、重启一个虚拟机（`vzctl restart 150`）或彻底从主机硬件驱动器中删除一个虚拟机（`vzctl destroy 150`），最后一个命令只有当先将虚拟机停掉后才会生效。

20.6 小 结

本章中，读者学到了如何在 `Linux` 环境中使用虚拟化技术，并讨论了使用虚拟化解决方案的 3 个不同方法。首先介绍了 `VirtualBox`，若不关注性能而是注重其易用性，出于演示的目的，那么这是一个好的选择。接下来，讨论了如何在主机上配置基于 `Xen` 和 `KVM` 的虚拟化环境。参考这些介绍，用户可以在网络中高效配置虚拟化和使用服务器资源。本章的最后部分还讨论了如何在主机环境中部署 `OpenVZ`。由于提出了虚拟容器的概念，`OpenVZ` 使用了完全不同的虚拟化解决方案。

[G e n e r a l I n f o r m a t i o n]

书名= L i n u x 系统管理大全

作者= (美) 特恩布尔, (美) 马托泰克著

页数= 8 2 0

S S 号= 1 2 7 5 3 6 1 0

出版日期= 2 0 1 0 . 1 2

封面
书名
版权
前言
目录

第一部分 入门

第 1 章	L i n u x 介绍
1 . 1	L i n u x 发行版
1 . 1 . 1	R e d H a t E n t e r p r i s e L i n u x
1 . 1 . 2	C e n t O S
1 . 1 . 3	T h e F e d o r a P r o j e c t
1 . 1 . 4	D e b i a n L i n u x
1 . 1 . 5	U b u n t u
1 . 1 . 6	G e n t o o
1 . 1 . 7	那么我们应该选择哪一种发行版呢？
1 . 1 . 8	本书涉及了哪些发行版呢？
1 . 2	选择硬件
1 . 3	支持的硬件
1 . 4	获取系统软件
1 . 5	获得支持
1 . 6	小结
第 2 章	安装 L i n u x
2 . 1	L i v e C D 与虚拟机
2 . 1 . 1	L i v e C D
2 . 1 . 2	虚拟机
2 . 2	R e d H a t E n t e r p r i s e L i n u x 的安装
2 . 3	U b u n t u 的安装
2 . 4	故障检修
2 . 4 . 1	诊断信息
2 . 4 . 2	重新安装
2 . 4 . 3	故障检修资源
2 . 5	小结
第 3 章	L i n u x 基础
3 . 1	准备开始
3 . 2	L i n u x 与 M i c r o s o f t W i n d o w s
3 . 2 . 1	G U I 桌面
3 . 2 . 2	命令行
3 . 3	远程访问
3 . 4	获得帮助
3 . 5	用户与组
3 . 6	服务与进程
3 . 7	软件包
3 . 8	文件与文件系统
3 . 8 . 1	文件类型与权限
3 . 8 . 2	链接
3 . 8 . 3	用户、组与所有权
3 . 8 . 4	大小和空间
3 . 8 . 5	日期与时间
3 . 9	文件操作
3 . 9 . 1	读文件
3 . 9 . 2	搜索文件
3 . 9 . 3	复制文件
3 . 9 . 4	移动与重命名文件
3 . 9 . 5	删除文件
3 . 9 . 6	链接文件
3 . 9 . 7	编辑文件
3 . 1 0	小结
第 4 章	用户和组
4 . 1	登入时发生了什么事？

- 4 . 2 用户与组操作
 - 4 . 2 . 1 s u d o 介绍
 - 4 . 2 . 2 创建用户
 - 4 . 2 . 3 创建组
 - 4 . 2 . 4 删除用户和组
 - 4 . 2 . 5 通过 G U I 管理用户和组
 - 4 . 2 . 6 密码
 - 4 . 2 . 7 密码时效
 - 4 . 2 . 8 停用用户
 - 4 . 2 . 9 存储用户和组数据
 - 4 . 2 . 1 0 配置 s h e l l 和环境
- 4 . 3 控制对主机的访问
- 4 . 4 s u d o 命令详解
- 4 . 5 小结

第 5 章 启动与服务

- 5 . 1 当主机启动时发生了什么事？
 - 5 . 1 . 1 B I O S
 - 5 . 1 . 2 引导程序
 - 5 . 1 . 3 操作系统
- 5 . 2 了解 G R U B 引导程序
 - 5 . 2 . 1 配置 G R U B
 - 5 . 2 . 2 使用 G R U B 菜单
 - 5 . 2 . 3 保护引导程序
- 5 . 3 启动之后发生了什么？
 - 5 . 3 . 1 配置 i n i t
 - 5 . 3 . 2 在运行级别之间移动
- 5 . 4 管理服务
 - 5 . 4 . 1 管理 R e d H a t 上的服务
 - 5 . 4 . 2 管理 U b u n t u 上的服务
 - 5 . 4 . 3 U p s t a r t ：一种新方式
- 5 . 5 关闭和重启主机
- 5 . 6 使用定时任务调度服务与命令
- 5 . 7 小结

第 6 章 网络与防火墙

- 6 . 1 网络与连网概论
 - 6 . 1 . 1 从接口开始
 - 6 . 1 . 2 在 G U I 下配置接口
 - 6 . 1 . 3 使用网络脚本配置网络
 - 6 . 1 . 4 添加路由与转发数据包
- 6 . 2 一般网络故障检修
 - 6 . 2 . 1 P i n g !
 - 6 . 2 . 2 M T R
 - 6 . 2 . 3 T C P / I P 1 0 1
 - 6 . 2 . 4 t c p d u m p 命令
 - 6 . 2 . 5 N e t c a t 工具
 - 6 . 2 . 6 d i g 它？
 - 6 . 2 . 7 其他故障诊断工具
- 6 . 3 N e t f i l t e r 与 i p t a b l e s
- 6 . 4 N e t f i l t e r / i p t a b l e s 是如何工作的？
 - 6 . 4 . 1 表
 - 6 . 4 . 2 链
 - 6 . 4 . 3 策略
 - 6 . 4 . 4 网络地址转换
 - 6 . 4 . 5 使用 i p t a b l e s 命令
 - 6 . 4 . 6 对 R e d H a t 主机上默认规则的解释
- 6 . 5 配置范例网络
 - 6 . 5 . 1 我们的配置
 - 6 . 5 . 2 其他防火墙配置工具
- 6 . 6 T C P W r a p p e r s

	6 . 7	小结
第 7 章		软件包管理
7 . 1		软件包管理介绍
7 . 2		Red Hat 上的软件包管理
	7 . 2 . 1	准备开始
	7 . 2 . 2	软件包更新器程序
	7 . 2 . 3	软件包管理器程序
	7 . 2 . 4	Red Hat 网络 (R H N)
	7 . 2 . 5	Yellowdog Updater Modified (Y u m)
	7 . 2 . 6	Red Hat 软件包管理 (R P M)
	7 . 2 . 7	通过源代码创建 R P M 软件包
7 . 3		U b u n t u 上的软件包管理
7 . 4		使用 S y n a p t i c 进行包管理
	7 . 4 . 1	添加软件
	7 . 4 . 2	移除软件
	7 . 4 . 3	管理软件源
	7 . 4 . 4	更新与升级
	7 . 4 . 5	使用更新管理器
7 . 5		使用 d p k g 包管理
	7 . 5 . 1	查看软件包细节
	7 . 5 . 2	安装软件包
	7 . 5 . 3	卸载软件包
7 . 6		编译源代码
	7 . 6 . 1	配置
	7 . 6 . 2	编译和生成
	7 . 6 . 3	安装
	7 . 6 . 4	卸载
	7 . 7	小结
第 8 章		存储管理与灾难恢复
8 . 1		存储器基础
	8 . 1 . 1	设备
	8 . 1 . 2	分区
	8 . 1 . 3	文件系统
	8 . 2	使用文件系统
	8 . 2 . 1	自动挂载
	8 . 2 . 2	检查文件系统的利用率
	8 . 3	R A I D
	8 . 3 . 1	R A I D 的类型
	8 . 3 . 2	创建阵列
8 . 4		逻辑卷管理
	8 . 4 . 1	创建卷与卷组
	8 . 4 . 2	扩充逻辑卷
	8 . 4 . 3	缩减逻辑卷
	8 . 4 . 4	用 G U I 管理 L V M
	8 . 5	故障恢复
	8 . 5 . 1	引导程序的问题
	8 . 5 . 2	磁盘故障
	8 . 6	小结
第二部分		让 L i n u x 为你工作
第 9 章		基础架构服务：N T P、D N S、D H C P 和 S S H
9 . 1		网络时间协议
9 . 2		域名系统
	9 . 2 . 1	根服务器
	9 . 2 . 2	查询域名服务器
	9 . 2 . 3	运行缓存 D N S
	9 . 2 . 4	权威 D N S 服务器
	9 . 2 . 5	动态 D N S
9 . 3		动态主机配置协议
	9 . 3 . 1	安装与配置

- 9 . 3 . 2 静态租约分配
- 9 . 3 . 3 动态DNS更新
- 9 . 3 . 4 手动修改DNS输入项
- 9 . 4 S e c u r e S h e l l
- 9 . 4 . 1 创建和分配密钥
- 9 . 4 . 2 使用SSH代理
- 9 . 4 . 3 调整SSH配置
- 9 . 4 . 4 执行快速又安全的文件传输任务
- 9 . 5 小结

第 1 0 章 邮件服务

- 1 0 . 1 电子邮件是如何工作的
- 1 0 . 1 . 1 发送一封电子邮件时发生了什么事？
- 1 0 . 1 . 2 电子邮件发送之后发生了什么事？
- 1 0 . 2 配置电子邮件
- 1 0 . 2 . 1 安装
- 1 0 . 2 . 2 启动P o s t f i x
- 1 0 . 2 . 3 了解P o s t f i x 配置
- 1 0 . 2 . 4 初始配置
- 1 0 . 2 . 5 测试P o s t f i x
- 1 0 . 2 . 6 选择邮箱格式
- 1 0 . 3 P o s t f i x 扩展配置
- 1 0 . 3 . 1 使用加密功能
- 1 0 . 3 . 2 身份验证
- 1 0 . 4 获得P o s t f i x 相关的帮助
- 1 0 . 5 防止病毒和垃圾邮件
- 1 0 . 5 . 1 与垃圾邮件作战
- 1 0 . 5 . 2 防病毒
- 1 0 . 6 配置I M A P 和P O P 3
- 1 0 . 6 . 1 I M A P
- 1 0 . 6 . 2 P O P 3
- 1 0 . 6 . 3 二者有什么区别？
- 1 0 . 6 . 4 在I M A P 和P O P 3 之间选择
- 1 0 . 6 . 5 D o v e c o t 介绍
- 1 0 . 7 虚拟域与虚拟用户
- 1 0 . 8 小结

第 1 1 章 W e b 服务和S Q L 服务

- 1 1 . 1 A p a c h e 网页服务器
- 1 1 . 1 . 1 安装和配置
- 1 1 . 1 . 2 访问控制
- 1 1 . 1 . 3 模块
- 1 1 . 1 . 4 文件和目录许可
- 1 1 . 2 M y S Q L 数据库
- 1 1 . 2 . 1 安装
- 1 1 . 2 . 2 测试服务器
- 1 1 . 2 . 3 基本的I n n o D B 优化技术
- 1 1 . 2 . 4 基本的M y S Q L 管理
- 1 1 . 3 安装网络站点
- 1 1 . 3 . 1 网络交流
- 1 1 . 3 . 2 w e b m a i l
- 1 1 . 3 . 3 配置S q u i r r e l M a i l
- 1 1 . 3 . 4 其他W e b 应用程序
- 1 1 . 4 S q u i d C a c h e
- 1 1 . 4 . 1 配置
- 1 1 . 4 . 2 客户端配置
- 1 1 . 4 . 3 透明性
- 1 1 . 5 小结

第 1 2 章 文件和打印共享

- 1 2 . 1 使用S a m b a 和N F S 共享文件
- 1 2 . 2 S a m b a

1 2 . 2 . 1	给 S a m b a 添加用户
1 2 . 2 . 2	向域中添加主机
1 2 . 2 . 3	S a m b a 所需的 I P 表规则
1 2 . 2 . 4	在 L i n u x 上挂载 S a m b a 的共享目录
1 2 . 2 . 5	使用 s y s t e m - c o n f i g - s a m b a 图形用户界面
1 2 . 2 . 6	资源
1 2 . 3	N F S 共享文件：L i n u x 到 L i n u x
1 2 . 4	管理文档
1 2 . 4 . 1	使用文档管理系统
1 2 . 4 . 2	开源 DMS K n o w l e d g e T r e e
1 2 . 4 . 3	安装 K n o w l e d g e T r e e
1 2 . 4 . 4	管理 K n o w l e d g e T r e e
1 2 . 4 . 5	处理文档
1 2 . 4 . 6	启动和停止 K n o w l e d g e T r e e 文档管理系统
1 2 . 4 . 7	带 S S L 的安全 K n o w l e d g e T r e e
1 2 . 4 . 8	资源
1 2 . 5	打印服务
1 2 . 5 . 1	C U P S
1 2 . 5 . 2	S a m b a 和打印服务：给桌面系统添加打印机
1 2 . 6	小结
第 1 3 章	备份和恢复
1 3 . 1	灾难恢复计划
1 3 . 2	备份过程
1 3 . 3	网络备份
1 3 . 4	使用 r s y n c
1 3 . 5	使用 B a c u l a
1 3 . 5 . 1	获取软件
1 3 . 5 . 2	配置 B a c u l a
1 3 . 5 . 3	使用 b c o n s o l e 管理 B a c u l a
1 3 . 5 . 4	使用 B a c u l a 备份数据库
1 3 . 5 . 5	介绍 B a t 控制台
1 3 . 6	小结
第 1 4 章	构建 V P N 网络
1 4 . 1	示例网络
1 4 . 2	O p e n V P N 介绍
1 4 . 2 . 1	安装 O p e n V P N
1 4 . 2 . 2	启动和停止 O p e n V P N
1 4 . 2 . 3	配置 O p e n V P N
1 4 . 2 . 4	用 O p e n V P N 发布总公司资源
1 4 . 2 . 5	为移动用户创建 V P N 连接
1 4 . 3	O p e n V P N 故障排除
1 4 . 4	小结
第 1 5 章	协作服务
1 5 . 1	Z i m b r a
1 5 . 2	安装 Z i m b r a
1 5 . 2 . 1	安装前提
1 5 . 2 . 2	下载和主机前期准备
1 5 . 2 . 3	安装 Z i m b r a
1 5 . 2 . 4	Z i m b r a 安装后续的配置菜单
1 5 . 2 . 5	更改防火墙
1 5 . 2 . 6	Z i m b r a 管理控制台
1 5 . 2 . 7	创建 C l a s s o f S e r v i c e
1 5 . 2 . 8	添加新用户
1 5 . 2 . 9	邮箱别名和邮件分发列表
1 5 . 2 . 1 0	添加资源
1 5 . 2 . 1 1	添加 Z i m l e t
1 5 . 2 . 1 2	添加一个 S S L 认证
1 5 . 2 . 1 3	全局设置
1 5 . 2 . 1 4	监控 Z i m b r a

- 1 5 . 3 使用 Z i m b r a
- 1 5 . 3 . 1 使用电子邮箱
- 1 5 . 3 . 2 使用 Z i m l e t
- 1 5 . 3 . 3 共享文件夹、地址簿、文档及其他资源
- 1 5 . 4 迁移已有的邮件服务
- 1 5 . 5 小结

第 1 6 章 目录服务

- 1 6 . 1 什么是 L D A P
- 1 6 . 2 总则
- 1 6 . 3 部署
- 1 6 . 4 安装
 - 1 6 . 4 . 1 R e d H a t 安装指导
 - 1 6 . 4 . 2 U b u n t u 安装指导
- 1 6 . 5 配置
 - 1 6 . 5 . 1 创建模式
 - 1 6 . 5 . 2 访问控制列表
 - 1 6 . 5 . 3 启动 s l a p d 守护进程
 - 1 6 . 5 . 4 设置 L D A P 客户端
- 1 6 . 6 L D A P 管理及其工具
 - 1 6 . 6 . 1 L D I F 文件与添加用户
 - 1 6 . 6 . 2 使用 L D I F 文件添加用户
 - 1 6 . 6 . 3 搜索 L D A P 树
 - 1 6 . 6 . 4 L D A P 中删除条目
 - 1 6 . 6 . 5 密码策略覆盖
 - 1 6 . 6 . 6 测试访问控制列表
 - 1 6 . 6 . 7 备份 L D A P 目录
 - 1 6 . 6 . 8 L D A P 账户管理：基于 W e b 的 G U I
 - 1 6 . 6 . 9 安装与配置
 - 1 6 . 6 . 1 0 为 L A M 添 A p a c h e 虚拟主机
- 1 6 . 7 与其他服务整合
 - 1 6 . 7 . 1 单点登录：集中 L i n u x 认证
 - 1 6 . 7 . 2 P A M 运行机制
 - 1 6 . 7 . 3 L D A P 和 A p a c h e 认证
 - 1 6 . 7 . 4 L D A P 与知识树 D M S 整合
- 1 6 . 8 小结

第 1 7 章 性能监控与优化

- 1 7 . 1 基本的健康状况检查
 - 1 7 . 1 . 1 C P U 利用率
 - 1 7 . 1 . 2 内存利用率
 - 1 7 . 1 . 3 磁盘空间
 - 1 7 . 1 . 4 日志
- 1 7 . 2 高级工具
 - 1 7 . 2 . 1 C P U 和内存利用率
 - 1 7 . 2 . 2 交换空间的使用
 - 1 7 . 2 . 3 磁盘存取
- 1 7 . 3 持续性能监控
 - 1 7 . 3 . 1 S N M P
 - 1 7 . 3 . 2 C a c t i
- 1 7 . 4 性能优化
 - 1 7 . 4 . 1 资源限制
 - 1 7 . 4 . 2 s y s c t l 和 p r o c 文件系统
 - 1 7 . 4 . 3 存储设备
 - 1 7 . 4 . 4 文件系统调整
- 1 7 . 5 小结

第 1 8 章 日志记录与监控

- 1 8 . 1 日志记录
 - 1 8 . 1 . 1 配置 s y s l o g
 - 1 8 . 1 . 2 启动与配置 s y s l o g 守护进程
 - 1 8 . 1 . 3 使用 l o g g e r 工具测试日志记录

	1 8 . 1 . 4	日志管理与轮替
	1 8 . 2	日志分析与关联
	1 8 . 2 . 1	S E C简介
	1 8 . 2 . 2	安装S E C
	1 8 . 2 . 3	运行S E C
	1 8 . 2 . 4	使用S E C
	1 8 . 2 . 5	S E C排错
	1 8 . 3	监控
	1 8 . 3 . 1	N a g i o s简介
	1 8 . 3 . 2	安装N a g i o s
	1 8 . 3 . 3	启动N a g i o s
	1 8 . 3 . 4	N a g i o s配置
	1 8 . 3 . 5	搭建N a g i o s控制台
	1 8 . 3 . 6	N a g i o s疑难解答
	1 8 . 4	小结
第 1 9 章	配置管理	
	1 9 . 1	自动配置
	1 9 . 1 . 1	在R e d H a t中使用C o b b l e r自动化配置
	1 9 . 1 . 2	在U b u n t u中进行自动化配置
	1 9 . 1 . 3	K i c k s t a r t和P r e s e e d
	1 9 . 2	配置管理
	1 9 . 2 . 1	P u p p e t简介
	1 9 . 2 . 2	安装P u p p e t
	1 9 . 2 . 3	配置P u p p e t
	1 9 . 2 . 4	连接第一个客户端
	1 9 . 2 . 5	创建第一个配置
	1 9 . 2 . 6	应用第一个配置
	1 9 . 2 . 7	为多个主机定义配置
	1 9 . 2 . 8	相关资源
	1 9 . 2 . 9	使用模板
	1 9 . 2 . 1 0	定义
	1 9 . 2 . 1 1	更多P u p p e t
	1 9 . 2 . 1 2	P u p p e t排错
	1 9 . 3	小结
第 2 0 章	虚拟化	
	2 0 . 1	虚拟化解决方案
	2 0 . 1 . 1	V i r t u a l B o x
	2 0 . 1 . 2	V M w a r e
	2 0 . 1 . 3	X e n
	2 0 . 1 . 4	K V M
	2 0 . 1 . 5	O p e n V Z
	2 0 . 2	使用V i r t u a l B o x
	2 0 . 2 . 1	安装V i r t u a l B o x
	2 0 . 2 . 2	使用V i r t u a l B o x创建虚拟机
	2 0 . 3	使用X e n安装虚拟机
	2 0 . 3 . 1	为使用X e n准备计算机环境
	2 0 . 3 . 2	创建X e n虚拟机
	2 0 . 3 . 3	管理X e n虚拟机
	2 0 . 3 . 4	自动启动X e n虚拟机
2 0 . 4	使用K V M安装虚拟机	
	2 0 . 4 . 1	为K V M虚拟化准备服务器：网络方面
	2 0 . 4 . 2	在U b u n t u服务器上设置K V M
	2 0 . 4 . 3	在K V M上安装一个W i n d o w s客户端操作系统
	2 0 . 4 . 4	在K V M上安装一个U b u n t u服务器客户端操作系统
	2 0 . 4 . 5	使用虚拟机管理器管理K V M虚拟机
2 0 . 5	使用O p e n V Z的虚拟化方案	
	2 0 . 5 . 1	安装
	2 0 . 5 . 2	创建O p e n V Z虚拟机
	2 0 . 5 . 3	O p e n V Z虚拟机基本管理

